12-2004

Distributed data mining using web services.

Vivek Mongolu 1977-
University of Louisville

DISTRIBUTED DATA MINING USING WEB SERVICES

By

Vivek Mongolu
B.S., Jawaharlal Nehru Technological University, 2000

A Thesis
Submitted to the Faculty of the
Speed Scientific School
University of Louisville
as Partial Fulfillment of the Requirements
For the Professional Degree

Master of Science

Computer Engineering and Computer Science Department
University of Louisville
Louisville, KY

December 2004

DISTRIBUTED DATA MINING USING WEB SERVICES

By

Vivek Mongolu
B.S., Jawaharlal Nehru Technological University, 2000

A Thesis approved on

Dec 1st 2004
(date)

by the following Reading and Examination Committee:

Anup Kumar, Thesis Director(CECS)

Rammohan K. Ragade (CECS)

Julius P. Wong (ME)

ACKNOWLEDGMENTS

I would like to thank Dr. Anup Kumar for his guidance and patience throughout this thesis process. I would also like to thank Dr.Rammohan Ragade for his comments and assistance. Thanks also to Dr.Julius P. Wong for his service on the reading committee.

I would also like to thank my parents and members of my family for their loving support throughout my career. Their hard work, dedication, and love made all of my successes possible.

ABSTRACT

DISTRIBUTED DATA MINING USING WEB SERVICES

Vivek Mongolu

Dec 1, 2004

With the increasing computational power and the decreasing cost of high bandwidth networks resulted in Distributed Systems. Distributed Data Mining is being used to analyze and monitor data in distributed systems. In the past, distributed technologies like Java RMI, CORBA were used for data mining but the result was a more tightly coupled system. Using web services a loosely coupled, interoperable distributed computing framework can be built.

The topic of this thesis is to investigate the use of web service in distributed data mining. This thesis involves the design, development and implementation of distributed data mining using web services as well as an in-depth look at technical aspects and future implication of such framework. A working framework will be created allowing a user to dynamically locate and run mining algorithms on data services or vice versa. The algorithm and data will be deployed as web services. The created web services will be registered at public registry servers. Two distributed data mining architectures will be presented, Data to Algorithm and Algorithm to Data. Finally, performance of the both the architectures will be compared with varying data using different public registry servers.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

FIGURE PAGE

CODE LISTINGS

LISTING PAGE

I. INTRODUCTION

With the advances in computing, communication and the rate at which data is being collected as resulted in distributed systems. Since data is now widely being distributed, building an efficient data-mining framework is becoming an increasingly scientific challenge. Web Services have been around for sometime now and have matured enough to be useful in building distributed data mining framework.

The purpose of this thesis is to show how Web Services can be used in building Distributed Data Mining framework. The user will have the ability to see all the Data Web Services that provide actual data and Algorithm Web Services that provide the mining algorithms and run Algorithm on Data or vice versa get back the result. This thesis shows how web services can be used to float algorithm as well as data in a distributed network.

Motivation

There are many reasons to investigate the use of web services in distributed data mining systems. Today most of the data mining is done in three-tier or four-tier system. In a three-tier system client sends the request to server, the server then runs mining algorithms on its local data. In a four-tier system, the server talks to another remote server, the remote server analyzes its local data. In both the systems, mining algorithms and the data are tied to the server making

it a closed system. An extendable and open distributed data mining system can be built by separating the algorithm and data.

Using web services the interaction between algorithm and data can be made dynamic. Also, the algorithm can dynamically locate and analyze the data making it a much more robust system. The main motivation for this thesis involves the realization that using web services a dynamic distributed data mining system can be built.

Thesis Contribution

- This thesis discusses the design and implementation of Distributed Data Mining Architecture using Web services. Two architectures will be presented and they are
  1. Algorithm to Data and
  2. Data to Algorithm.
- Algorithm to Data: In this architecture, the algorithm is taken to different data nodes and then executed at these data nodes and the result is returned to the client.
- Data to Algorithm: In this architecture, the data is taken to the algorithm nodes. The algorithm is then executed with the coming data and the result is returned to the client.
- Performance comparison: Finally, the performance of the above mentioned architectures is compared using different registry servers.

Thesis Organization

Chapter II presents an introduction to different technologies used for this thesis. Chapter III focuses on the design of the distributed data mining architecture. Chapter IV discusses the implementation of the design of distributed data mining architecture using web services. This chapter will present the more technical aspects, basic operation of the final system along with screen shots. Chapter V compares the performance of the two architectures. Finally, Chapter VI gives a brief conclusion and recommendations for future work.

II. RELATED WORK

Distributed Data Mining

Mining distributed data by paying careful attention to the distributed resources is called Distributed Data Mining [3]. In a Distributed Data mining system the databases, data, archives, and algorithms are located at different sites. The algorithms or queries are passed to multiple sites and responses from these sites are combined and the final result is presented to the user. From the end user perspective, it appears like all these resources are at one single site.

Web Service

A Web service [4] is a service offered by one application to another via the World Wide Web (WWW). Web services uses one or more of the following XML based standards

- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal Description Discovery Integration (UDDI)

A web service makes itself available by describing itself in a Web Services Description Language (WSDL) document. WSDL document is a XML document

that has all the information about web service, including its name, the operations it supports, parameters for those operations and the location where it is running. For service consumers or clients to locate the web service, web service provider has to publish the web service at registry server. Universal Description, Discovery, and Integration (UDDI) is a standard protocol to publish or to find web services. The description part in UDDI is for service provider to publish details about their organization and web services they provide. The discovery part in UDDI is for service consumers or clients to find these services. Service consumer or client invokes web service using Simple Object Access Protocol (SOAP). SOAP is a lightweight XML protocol used for information exchange between heterogeneous systems.

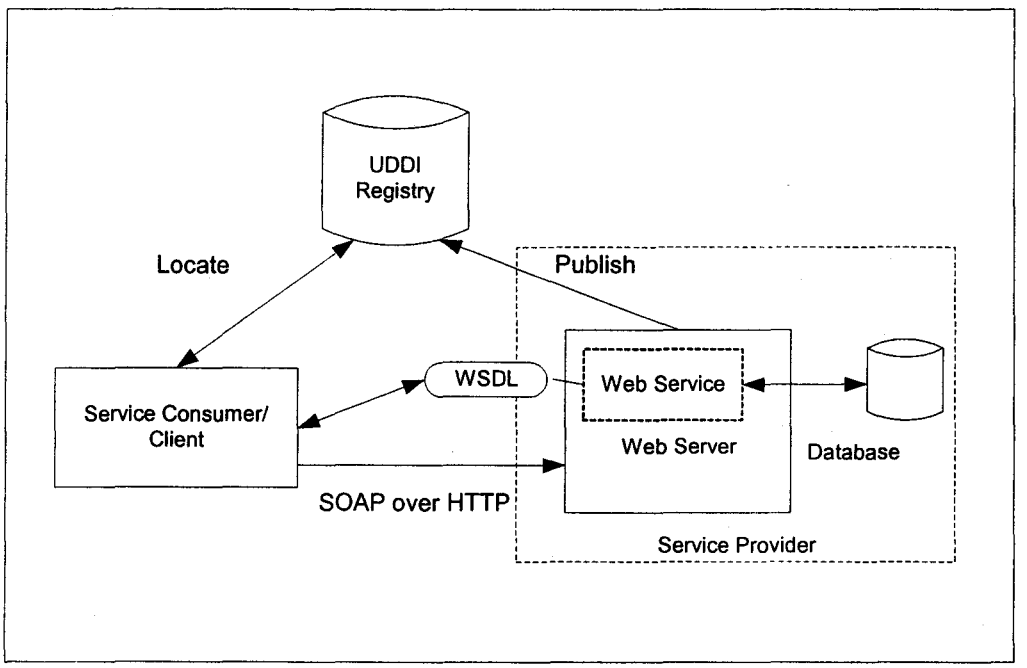

Figure 1 – Web Service architecture

Figure 1 shows simple web service architecture. Service provider registers the service in UDDI Registry. Client finds the service, gets the WSDL for the service. Using WSDL, client constructs a web service call and invokes the service.

Advantages of Web Services:

- Interoperability: Client written in Java can invoke a service written in .NET or vice versa.
- Ubiquity: Once the web service is registered in a registry server, any organization can use it.
- Loosely coupled: Web service doesn't depend the underlying system architecture.
- Industry standards: Web service supports industry based XML standards.

Creating a Distributed Data Mining architecture using Web services requires a lot of new technologies. Java Web Services Developer Pack (JWSDP) from Sun Microsystems solves many of the issues. It is through this pack that this thesis was created.

JWSDP

Java Web Services Developer Pack [4] is bundle of all the technologies that simplifies developing and deploying web services and web service clients. This

thesis uses many of the JWSDP's features. JWSDP is a collection of API's. The two used for this thesis are JAXRPC and JAXR.

JAXRPC

JAX-RPC [5] stands for Java API for XML-based Remote Procedure Calls. As the name implies, it enables building web services and web service clients that use remote procedure calls.

JAXR

JAXR [6] stands for Java API for XML Registries. It provides an API for querying and accessing web services from XML Registries and publishing web services to XML Registries.

Servlets

Servlets [7] are programs that run on a web server and build web pages dynamically. Servlets add functionality to Web Server just like applets add functionality to browser. Servlet is Java Class that implements the Servlet Interface. Servlets support different protocols but it is mostly used for HTTP specific services.

Servlets support request-response model. In a request/response model, a client sends a request to a server and the server responds by sending back a reply. For instance, a servlet may present an HTML form to the user. The user fills in the fields and clicks the submit button which posts the data to a Servlet (request). The Servlet processes the data by converting into appropriate data

types and then may store the data into Database. It responds back with a "Thank you" message (response). Servlets are the foundation for JSP and JSTL.

JSP and JSTL

JavaServer Pages separates the dynamic part of the page from static HMTL part. This enables rapid development of web-based applications. Page designers can concentrate on the overall page layout and developers can just worry about the dynamic content or the page logic. Dynamic content is produced with special tags. One of them is a Scriptlet, which start with "<%" and end with "%>".

When the JSP page is requested for the first time, JSP Engine converts the JSP page into Servlet Class and then compiles it. This is called Translation Phase and it happens only once unless the JSP page is altered. Request processing phase runs the resulting Servlet class to produce response for the client.

JSP syntax is similar to XML syntax. The page logic comes from Java Beans. Java Bean is a class that adheres to certain property and event interface conventions. JSP technology is being extended to develop simple tag libraries that are much cleaner and simpler. JavaServer Pages Standard Tag Library [8] is one of the tag libraries from Sun.

JavaServer Pages Standard Tag Library (JSTL) is a collection of standard tags that handle the most needed tasks for the JSP pages. The resulting JSP looks lot cleaner and can be easily maintained.

Listing 1 shows a sample JSP page using JSTL. It prints out "Hello World" and the current time at the Server in HTML to the user.

```
<%@ page import="java.util.Date"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
    <TITLE>Sample JSTL </TITLE>
  </HEAD>
  <BODY>
    <%
      Date date = new Date();
    %>
    <c:set var="hello" value="Hello World"/>

    <H1>
      <c:out value="${hello}"/>
    </H1>
    <h4>
      Time at Server is : <%=date%>
    </h4>

  </BODY>
</HTML>
```

Listing 1 – Sample JSP page

Web client part of the thesis is written using JSP, Java Beans and JSTL.

JDOM

JDOM [9] is a simple Java API used for creating and parsing XML. JDOM was used for parsing XML related data.

Tomcat and Ant

JWSDP comes with a Tomcat Server and has support for ANT. Both are from Apache. Together, they allow simple and quick deployment of web services.

ANT is an open-source Java-based build tool, which uses XML based configuration files. Tomcat server is a Java based Web Application container that runs JavaServer Pages (JSP) and Servlets in Web applications.

Registries

An XML Registry is an infrastructure that enables the building, deployment and discovery of Web services. It is a neutral third party that facilitates dynamic and loosely coupled business-to-business (B2B) interactions. Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web and make their systems interoperable for e-commerce.

III. DESIGN OF DISTRIBUTED DATA MINING ARCHITECTURE

The proposed architecture for Distributed Data Mining Framework uses Web Services. In this architecture, algorithms as well as data are web services. Algorithm web service returns the algorithm requested and Data web service returns the data in XML format by querying its database. For simplicity, MS Access database was used.

These services are registered in a UDDI Registry Servers so that a client can query the registry to access these services. Two data services and two algorithm services were used since the purpose of this thesis is to show how web services can be used in building distributed data mining and not creating complex data mining algorithms.

JAX-RPC was used to create Algorithm and Data web services. In JAX-RPC a remote procedure call is represented by SOAP, an XML-based protocol. JAX-RPC API hides much of the complexity and the created web services are platform and language independent.

Three registry servers were picked to show how these services interact, Local Registry Server which comes bundled with JWSDP, Microsoft test UDDI Registry and IBM test UDDI Registry.

Figure 2 shows that the Data web services and Algorithm web services are registering at the registry server. Client queries the registry server for Algorithm and Data web services. Registry server returns back all the information it has regarding the services. This information is sufficient to make a web service call and get the result back.

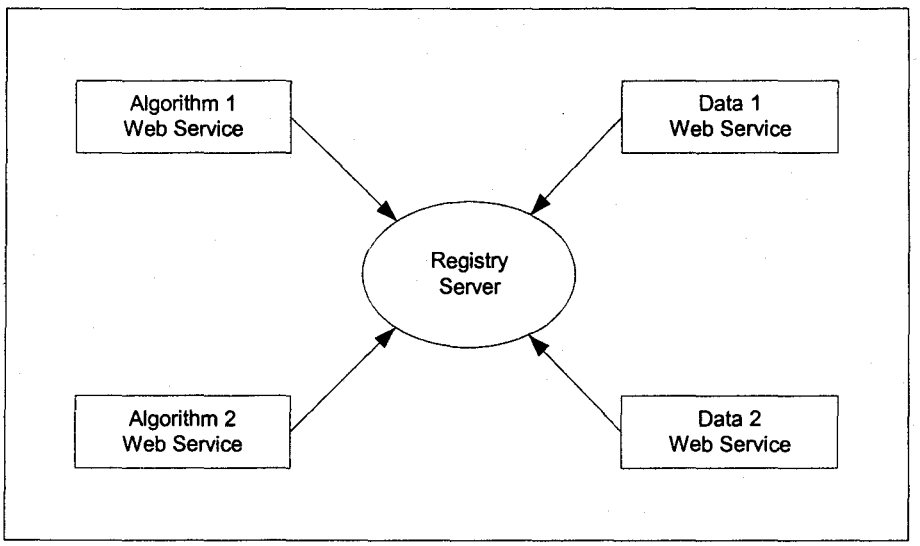

Figure 2 – Web Services registering at Registry server

Given this design two architectures are possible, algorithm can be taken to data or data can be taken to algorithm. The following sections discuss these two architectures in detail.

Algorithm to Data

In this architecture, the algorithm is taken to different data service nodes. The data service runs the algorithm with its data and returns back the result. This is an efficient architecture since in most cases data is huge.

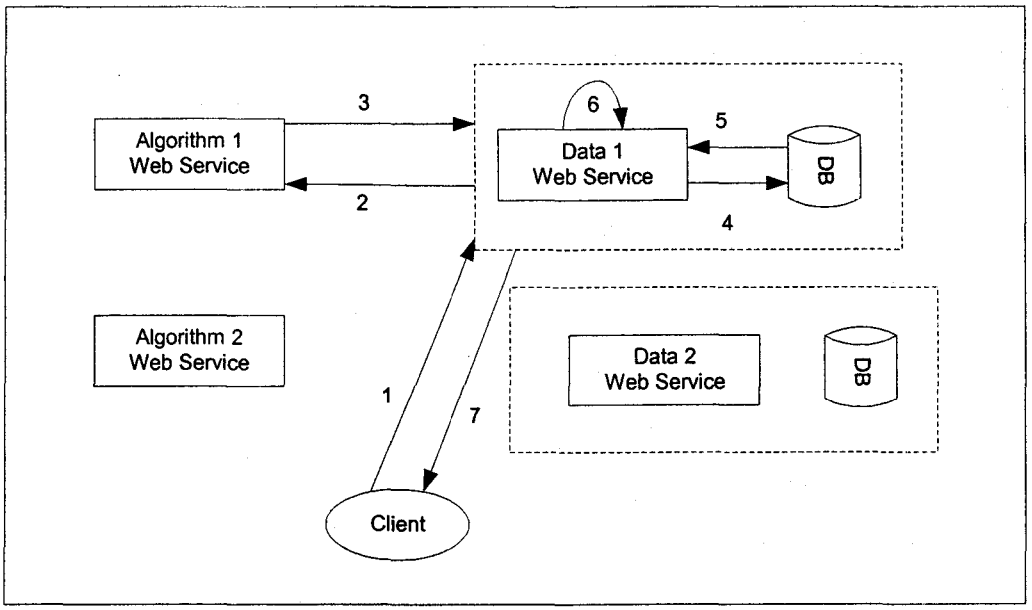

Figure 3 – Algorithm to Data architecture

Data web services are designed in such a way that the service knows how to interact with Algorithm web services. All the client has to do is pass information of Algorithm web service to Data web service. If Algorithm 1 needs to be executed at Data 1 then Algorithm 1 information obtained from registry server must be passed to Data 1.

In Figure 3, the client is calling Data 1 web service and passing the information required for Data 1 to call Algorithm 1 web service (1). Data 1 web service calls the Algorithm web service (2). Data 1 web service is now a client trying to access Algorithm 1 web service. Algorithm 1 web service will return the requested algorithm back to Data 1 Web Service (3). After Data 1 gets the algorithm, it queries the DB to get the data (4,5), it then executes the algorithm with this data and return back the result to client (7).

Data to Algorithm

In this architecture, the data is taken to Algorithm services. Algorithm service runs the algorithm with the coming data and returns back the result. Data Service wraps the data as an XML document and sends it to the Algorithm service. Algorithm service parses the XML document and retrieves the data. It then executes the algorithm with the retrieved data.

Algorithm web services are designed in such a way that the service knows how to interact with Data web services. Client has to pass the information to the Algorithm web service so that the Algorithm web service can call the Data web service. If Data 2 needs to be executed with Algorithm 1 then client will pass the Data 2 information obtained from registry server to Algorithm 1.

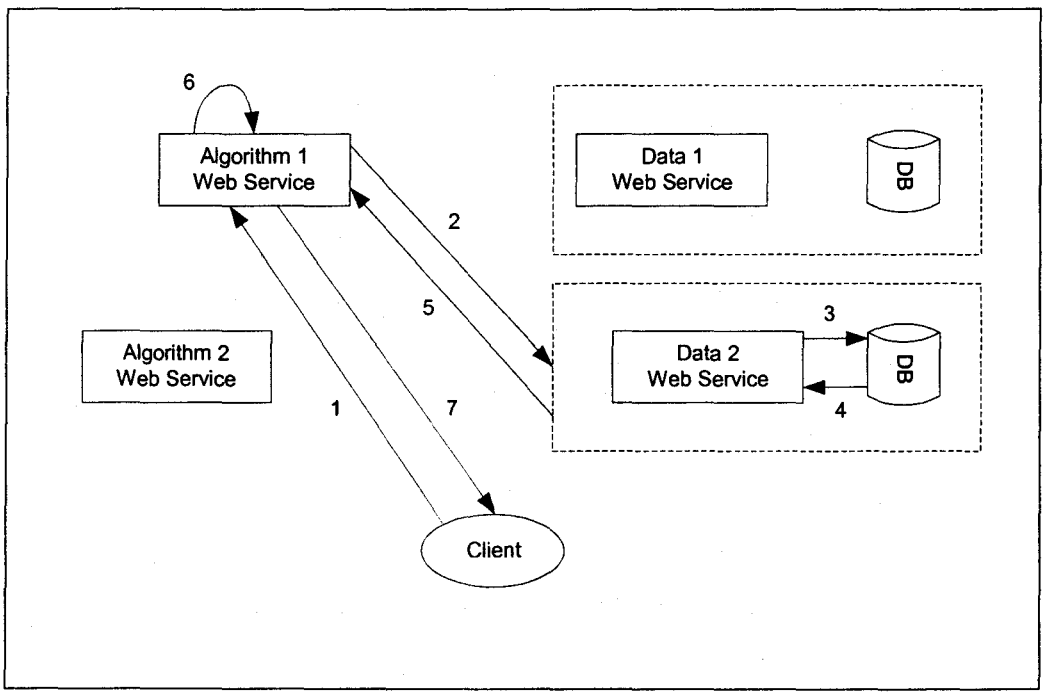

Figure 4 – Data to Algorithm architecture

In Figure 4, client calls the Algorithm 1 web service (1). Algorithm 1 web service requires the information to call Data 2 web service, which is passed by the client. Algorithm 1 web service makes a web service call to Data web service (2). Algorithm 1 web service is now a client accessing Data 2 web service. Data 2 web service queries its DB and wraps the data in the form of XML document (3,4) and returns this to Algorithm 1 web service (5). Algorithm 1 web service parses the XML document, retrieves the data and runs this data against the algorithm (6). The result is returned back to the client (7). Both the above-discussed architectures are implemented in this thesis

There were numerous obstacles encountered while building a Distributed Data Mining Framework. The main problem was with the Algorithm web service. The purpose of the Algorithm web service was to return the requested algorithm which is a ". java" file. It wasn't possible to return the file as it is over the network, so to deal with this issue it was agreed Algorithm web service would return it in the form of a DataHandler Object. A DataHandler object provides a standard interface to the data available in many different formats. It allows simple stream to string conversions.

The other problem was with Data web service. Data web service has to return the data in format so that Algorithm web service can understand it and use it. Since XML is extensively used in data exchange, it was agreed that Data web service would return data in XML format. A parser was written for the Algorithm web service to parse the XML document to get the required data.

Web Interface

The purpose of having a web interface is to present the user with the option to pick a Registry Server and then showing all the web services registered at that registry. Web Services interaction (as discussed earlier) includes algorithm to data or data to algorithm and also the interaction with the DB at the Data service end.

This entire interface was designed using JSP-Java Beans and JSTL. The web module was deployed to the Tomcat Web Server, which comes bundled with JWSDP.

Command Line Interface

It's a user interface in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system. Most of the beans designed for web interface are reused for command line interface. This interface also presents the user with the available registries and an option to pick one, and then shows all the registered web services. Interaction between web services is same as in web interface.

IV. IMPLEMENTATION OF DISTRIBUTED DATA MINING ARCHITECTURE

Implementation of Distributed Data Mining involved creating various web services, web service clients and registry client. The implementation was followed as discussed in the last chapter and working Distributed Data Mining using Web Services framework with control through web interface and command line interface, was the result.

Web Services Implementation

JAX-RPC was used to create Algorithm and Data Web Services. JAX-RPC is an API for building web services and clients that use remote procedure calls (RPC). RPC is often used in distributed client/server model and it enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML based protocol such as SOAP. SOAP defines envelope structure, body, message, conventions for representing remote procedure calls and responses. The JAX-RPC API hides all the complexity related to SOAP messages. JAX-RPC takes care of generating and parsing the SOAP messages. It is the JAX-RPC runtime system that converts the API calls and responses to and from SOAP messages.

There are three basic steps for creating a JAX-RPC based web service

1. Code the service endpoint interface and implementation class. Interface will contain method signatures and these methods will be exposed as Web services. Implementation class defines the functionality for these methods.
2. Build, generate, and package the files required by the service into Web Archive (WAR) file. WAR file is a package of the entire web application including configuration information and custom tag libraries.
3. Deploy the WAR file. Ant was used to build and deploy the WAR file to Tomcat Server.

As the architectures showed in the last chapter, each of the web service is also a client. Using JAX-RPC three types of clients can be created

1. Static stub
2. Dynamic proxy
3. Dynamic invocation interface (DII)

Static stub and Dynamic proxy rely on run time classes generated by JAX-RPC system. In contrast, DII client can make remote procedure call without even knowing the signature of the remote procedure or the name of the service until runtime. Since the name of the service and parameters required to make a call to remote procedure are dynamically passed from client to web service, DII was used.

Algorithm Web Services

The architectures discussed in last chapter showed that Algorithm web services have two functions

- return the requested algorithm.
- get the data from Data web service, run this data with its algorithm and return the result.

For these reasons, each of the Algorithm service interfaces have two methods, one that returns the algorithm as a DataHandler object and the other runs this algorithm with the data coming from Data web services. The data returned is in XML format and needs to be parsed to get the actual data.

For this thesis two simple algorithm web services were created, Mean Variance Algorithm web service and Outlier Algorithm web service. Mean Variance Algorithm web service returns the mean variance algorithm and Outlier Algorithm web service returns the outlier algorithm. The mean variance algorithm calculates the mean and the variance for a given data set. The outlier algorithm shows all the outliers for a data set using the combined mean and combined variance. The combined mean and combined variance is calculated using mean and variance obtained from two different data sets. These algorithms are Java Class files that can be compiled and executed.

Mean Variance Algorithm Web Service

First step in implementing Mean Variance Algorithm web service was to create service interface. Listing 2 shows this interface

```
public interface AlgorithmInter extends Remote
{

    public DataHandler getAlgorithmFile(String name)
        throws RemoteException;

    public String getDataRunMeanVariance(String qNameService,
                                          String qNamePort,
                                          String endPoint,
                                          String namespace)
        throws RemoteException;

}
```

Listing 2 – Mean Variance Algorithm Web Service Interface

Mean Variance Implementation Class provides the functionality for these methods. The getAlgorithmFile() method reads the file from hard disc and converts it into a DataHandler Object and this object is returned. The getDataRunMeanVariance() method calls the Data web service using the arguments to get the data. Method then compiles and executes mean variance algorithm with this data. The result is returned as a String.

Outlier Algorithm Web Service

First step in implementing Outlier Algorithm web service was to create service interface. Listing 3 shows this interface. Outlier Implementation Class

provides the same functionality as discussed in Mean Variance Implementation Class.

```
public interface AlgorithmInter extends Remote
{

        public DataHandler getAlgorithmFile(String name)
            throws RemoteException;

        public String getDataRunOutliers( String qNameService,
                                          String qNamePort,
                                          String endPoint,
                                          String namespace,
                                          String combinedMean,
                                          String combinedVariance)
            throws RemoteException;

}
```

Listing 3 – Outlier Algorithm Web Service Interface

Data Web Services

The architectures discussed in last chapter showed that Data web services have two functions:

- return the data in XML format to Algorithm web service
- It needs to get the algorithm from Algorithm web service and then run this algorithm with its data and return the result.

Since two algorithm web services were used, the Data service interfaces have three methods, one that returns the data and the other two invoke the respective algorithm web services to get the algorithms. In both the cases Data web service queries the database table to get the data. For simplicity the data is

stored in a single table. Two Data web services are deployed, Data 1 web service and Data 2 web service.

To run the algorithm with its data, Data web service needs to call Algorithm web service. Data web service makes a call to Algorithm web service and gets the algorithm, then creates the file and saves it to the hard disc. This is a java file, which has to be compiled and executed. Runtime and Process class are used to do the compiling and executing.

To return the data to Algorithm web service, Data web service places this data in a XML format. Here we are assuming that both parties agree on certain schema, so that Algorithm web service knows how to parse the XML document and retrieve the actual data. For creating and parsing XML document, JDOM was used.

Data 1 and Data 2 Web Service

Data 1 and Data 2 service interface and implementation class are exactly same but they are deployed as different services running on different machines. Service interface has three methods. Listing 4 shows this interface. Implementation Class defines the functionally for these methods. The getData method returns the data to Algorithm web service. The getMeanVariance and getOutlier method calls the respective Algorithm web services to get the algorithm and later executes these algorithms with its data. The arguments to these methods identify the algorithm service name and the endpoint where it's running.

```
public interface DataInterface extends Remote
{

        public String getMeanVariance(String endPoint,
                                      String qNameService,
                                      String qNamePort,
                                      String namespace)
            throws RemoteException;

        public String getOutliers(String endPoint,
                                  String qNameService,
                                  String qNamePort,
                                  String namespace,
                                  String combinedMean,
                                  String combinedVariance)
            throws RemoteException;

       public String getData(String table)
            throws RemoteException;

}
```

Listing 4– Data Web Service Interface

JDBC

JDBC (Java Database Connectivity) is a standard API for database connectivity between Java and various available databases. Distributed Data Mining Framework uses Microsoft Access Database. JDBC is used in web interface client, command line interface and also in Data web services. In web interface and command line interface, it's used to get all the registries from database. Registries and Registry classes serve this purpose. Their source code can be found in Appendix A. In Data web services, its used to query the database to get the actual data.

Listing 5 shows a simple way to connect to database and execute queries. Connection is made to the database using the datasource name "thesis" which is setup in ODBC in Windows.

```
Connection connection = null;
Statement statement = null;
ResultSet rs;

try {

    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection=DriverManager.getConnection("jdbc:odbc:thesis"," "," ");
    statement = connection.createStatement();

    rs = statement.executeQuery("SELECT * FROM TABLE_A");

}catch(SQLException e){
    System.out.println("SQLException : " + e.toString());
}
```

Listing 5 – Sample JDBC Usage

Registries

After coding the web services, they were deployed to a Tomcat server. Algorithm web services were deployed to one machine and the Data web services were deployed to another machine. Next these web services were registered at three registry servers

- Local Registry server that ships with JWSD
- IBM Test UDDI Registry
- Microsoft UDDI Registry Serve

The process of registering is very simple.

The next thing to implement was a class that would query the registry server to pull information regarding the web services. JAXR API was used since it provides an easy to use API to access various XML registry servers. The result was JAXQuery class. Its source code can be seen in Appendix A. A method called getDataMiningServices was created to query the registry server. This method takes URL of the Registry Server to query as a first argument and the organization name to search for as a second argument. When registering the Algorithm web services, the organization name is entered as "algorithmwebservice" and for Data web services, the organization name is entered as "dataservices". This method is invoked twice to get algorithm and data web services. For each service found in the registry, it gets the service name, service description, service binding, URI and builds JAXQueryResult object with this information. The source code for JAXQueryResult can be found in Appendix A. It then creates and returns a List of JAXRQueryResult objects. Figure 5 shows the class diagrams for JAXRQuery and JAXRQueryResult.

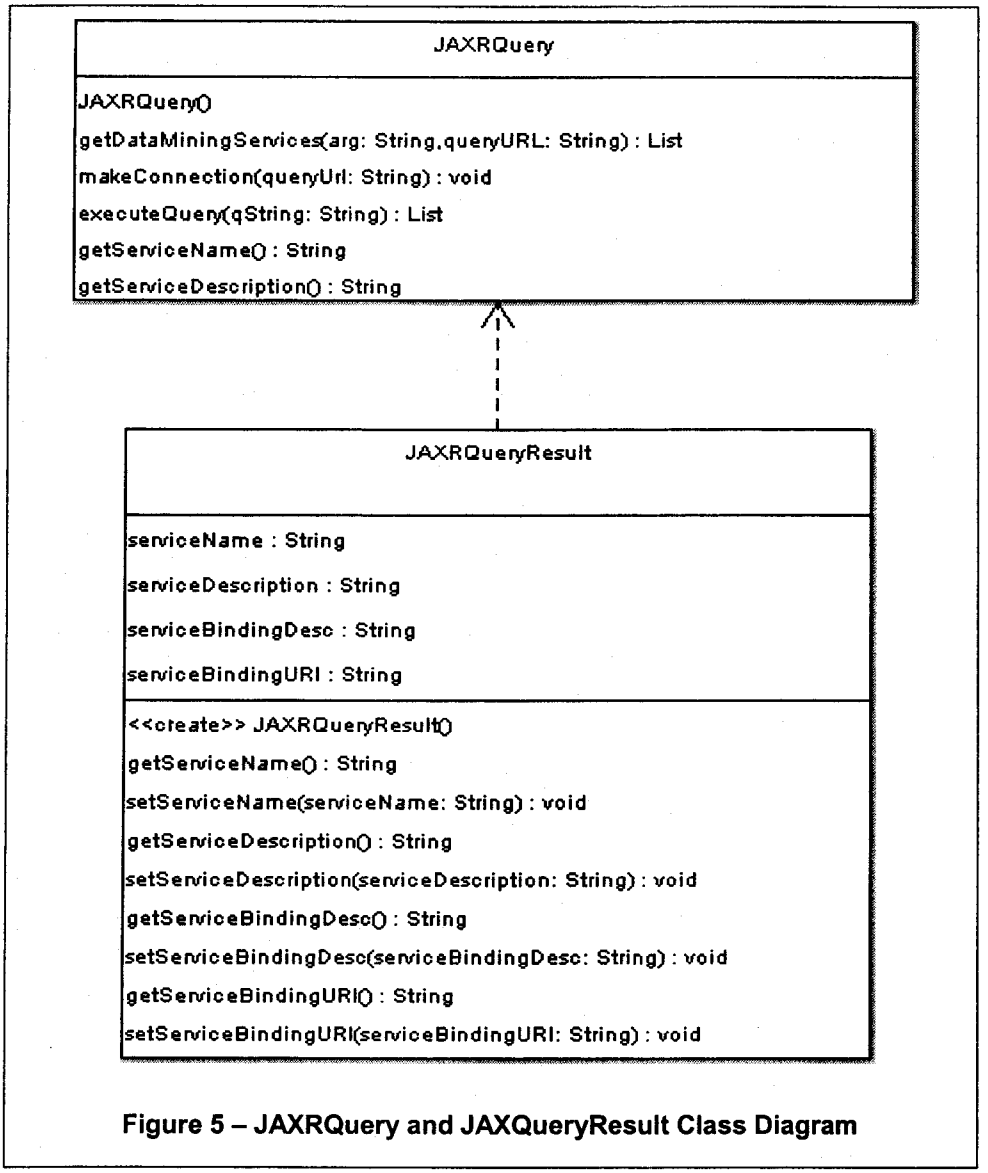

Figure 5 – JAXRQuery and JAXQueryResult Class Diagram

WSDL Parser

The URI in the JAXRQueryResult object is the URL of the running web service and it is the most important part of web service. Using the URI of the service, WSDL of the service can be known. Web Service Definition Language (WSDL) is an XML document that specifies the location of the service and the operations or methods that the service exposes. In order to retrieve all this information from WSDL, a parser class called WSDLParser was written. WSDLParser parses the input WSDL File and constructs the WSDLObject. WSDLObject saves the service name, service port, service endpoint and the name space of the service. This WSDLObject information is very useful. When the client invokes Algorithm web service, client has to pass the Data web service information to Algorithm web service so that the Algorithm web service can invoke the Data web service. The required Data web service information is in the WSDLObject, this information can be easily passed to Algorithm web service. The source code for WSDLParser and WSDLObject can be found in Appendix A. Figure 6 shows the class diagram of WSDLParser and WSDLObject.

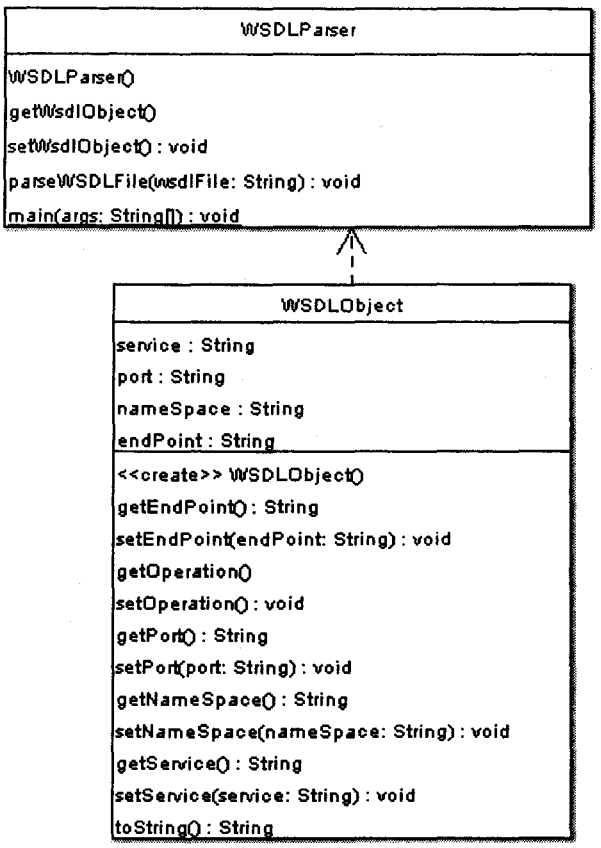

Figure 6 – WSDLParser and WSDLObject Class Diagram

Sample Working model

Figure 7 shows a sample-working model. First all the web services register with Registry Server (1). Client can be either the web interface or command line interface. Client queries the Registry Server for Algorithm web services and Data web services (2). Registry Server returns back the registered services. Client uses JAXRQuery Class to query the registry server. JAXRQuery Class constructs a JAXRQueryResult object for each service found in the registry. Since two algorithm and two data web service were registered, client now as two

Algorithm JAXRQueryResult objects and two Data JAXRQueryResult objects. URI is one of the properties of JAXQueryResult object using, which, the client gets the WSDL file. Four WSDL files are parsed with WSDLParser class (3). WSDLParser class constructs a WSDLObject for each WSDL file parsed. Client now as two Data WSDLObjects and two Algorithm WSDLObjects (4). These WSDLObjects have full information regarding the web services. These objects are used for interaction between the web services. The figure shows one such case. Client wants the data from Data 1 Web service to be run at Mean Variance Algorithm Web service end. Client invokes the Mean Variance algorithm web service passing to it Data 1 web service (5) information. Mean Variance web service uses this information to make an RPC web service call to Data 1 web service (6). Data 1 web service queries its database using JDBC (7). It then formats the data in the form a XML document (8). This is returned to Mean Variance Algorithm web service (9). Mean Variance web service then parses the XML document and runs the retrieved data through its algorithm (10) and the result is returned back to client (11), which is displayed for analysis (12). Take another case where in the client wants the data from Data 2 web service to be executed with the outlier algorithm from Outlier Algorithm web service. Since the client has Data 2 WSDLObject and the Outlier Algorithm WSDL object, client can invoke Data 2 web service. Client will pass the Outlier Algorithm web service information to the Data 2 web service. Data 2 web service uses this information and makes an RPC web service call to Outlier Algorithm web service. Outlier Algorithm web service responds back with outlier algorithm. Data 2 web service

queries its database, then runs the data with the outlier algorithm and the result is sent back to the client for analysis.

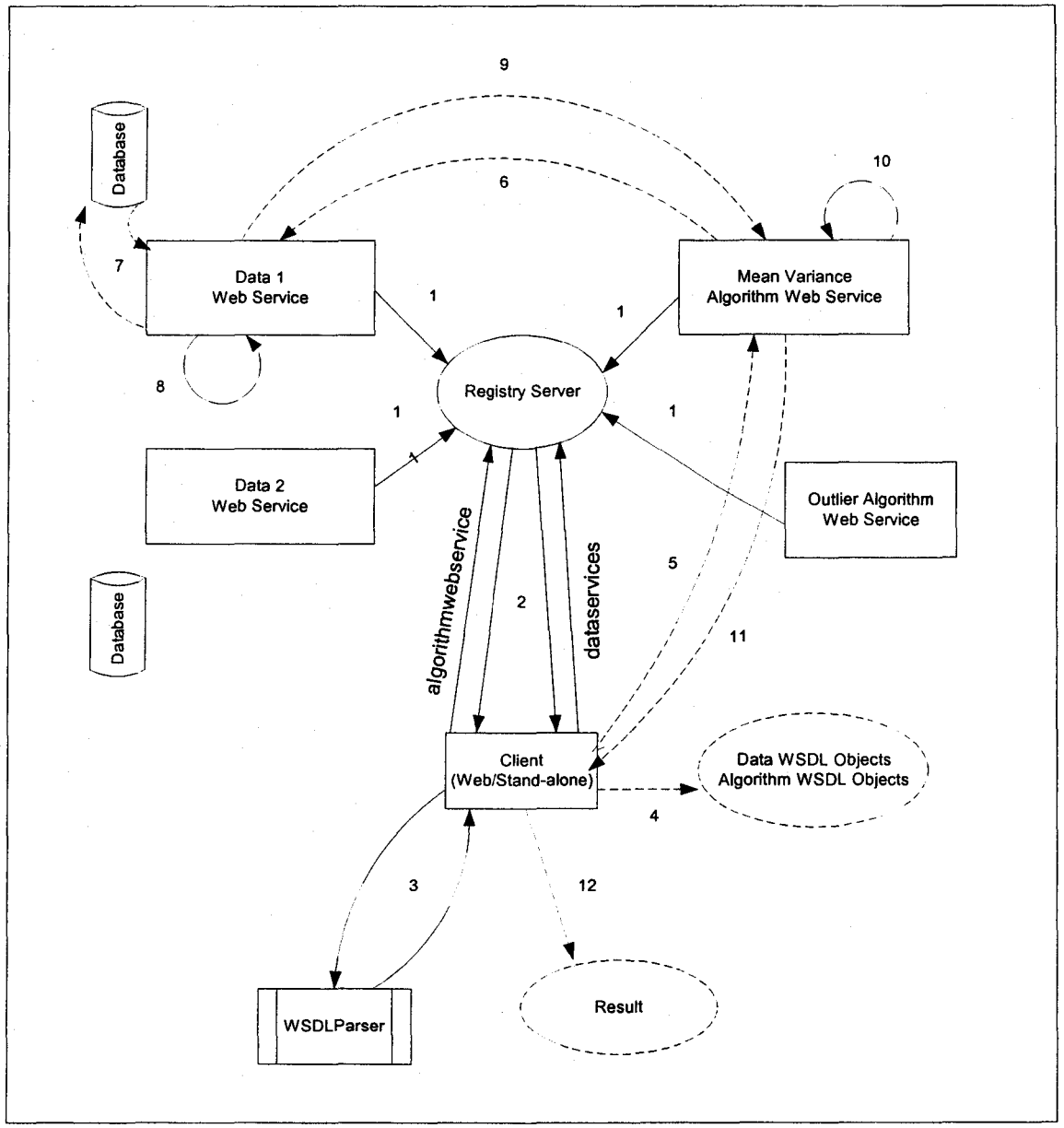

Figure 7 – Sample working model

All possible web service interactions are implemented. If the user selects Algorithm to Data, then first Mean Variance Algorithm is taken to Data 1 and Data 2, the algorithm is executed at both the nodes and the result is returned back to client. Using the results, client calculates the combined mean and combined variance. Next Outlier Algorithm is taken to Data 1 and Data 2, algorithm is executed at both the nodes and the result is returned back to client. If the user selects Data to Algorithm, then first mean variance algorithm is executed with data coming from Data 1 and Data 2. Client then calculates combined mean and combined variance. Next outlier algorithm is executed with data coming from Data 1 and Data 2 and the result is returned back to client.

Screenshots

The result of the implementation is a working framework. Its usage is described along with screenshots.

Main Web Interface

The main page of the system is a simple front page that shows all the registries available and an option to pick one. Figure 8 and Figure 9 shows the screen shot of the front page of the framework.

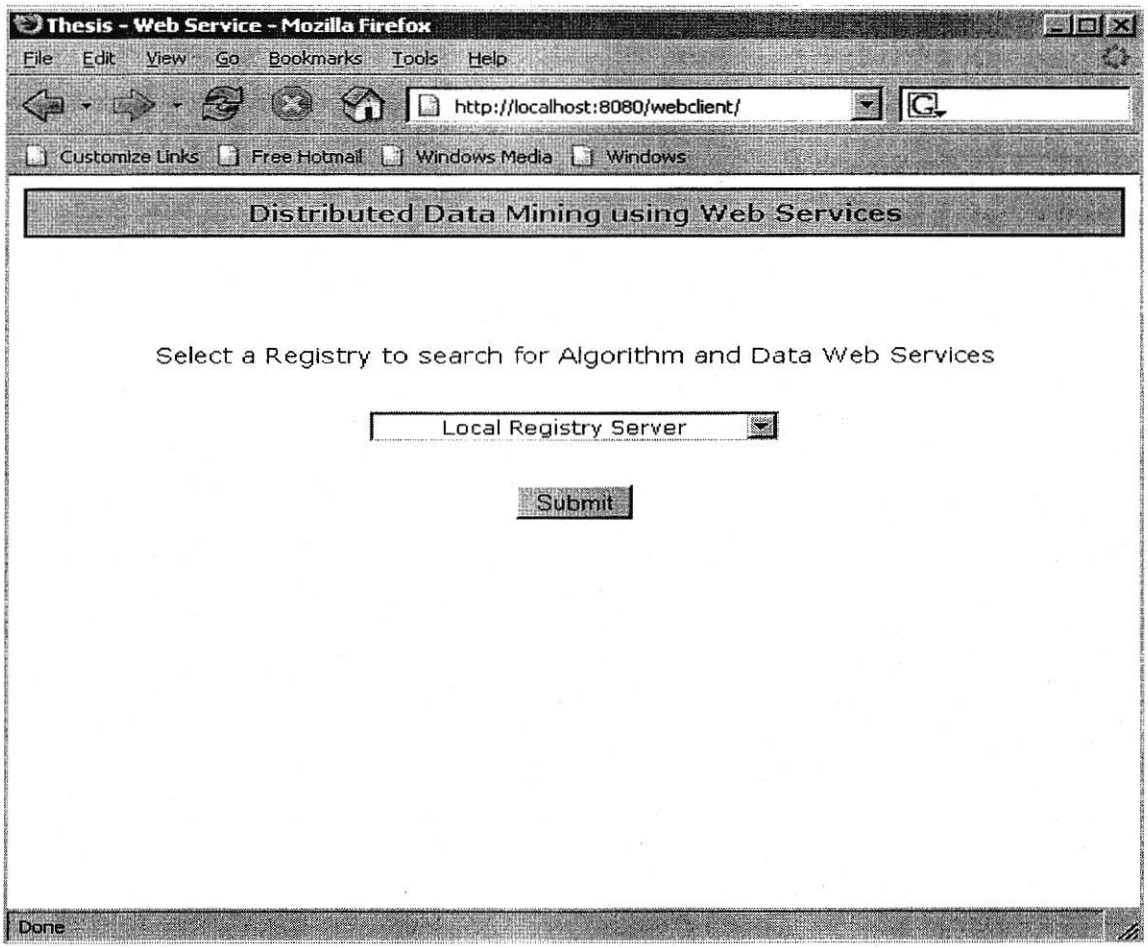

Figure 8 – Screenshot of main page

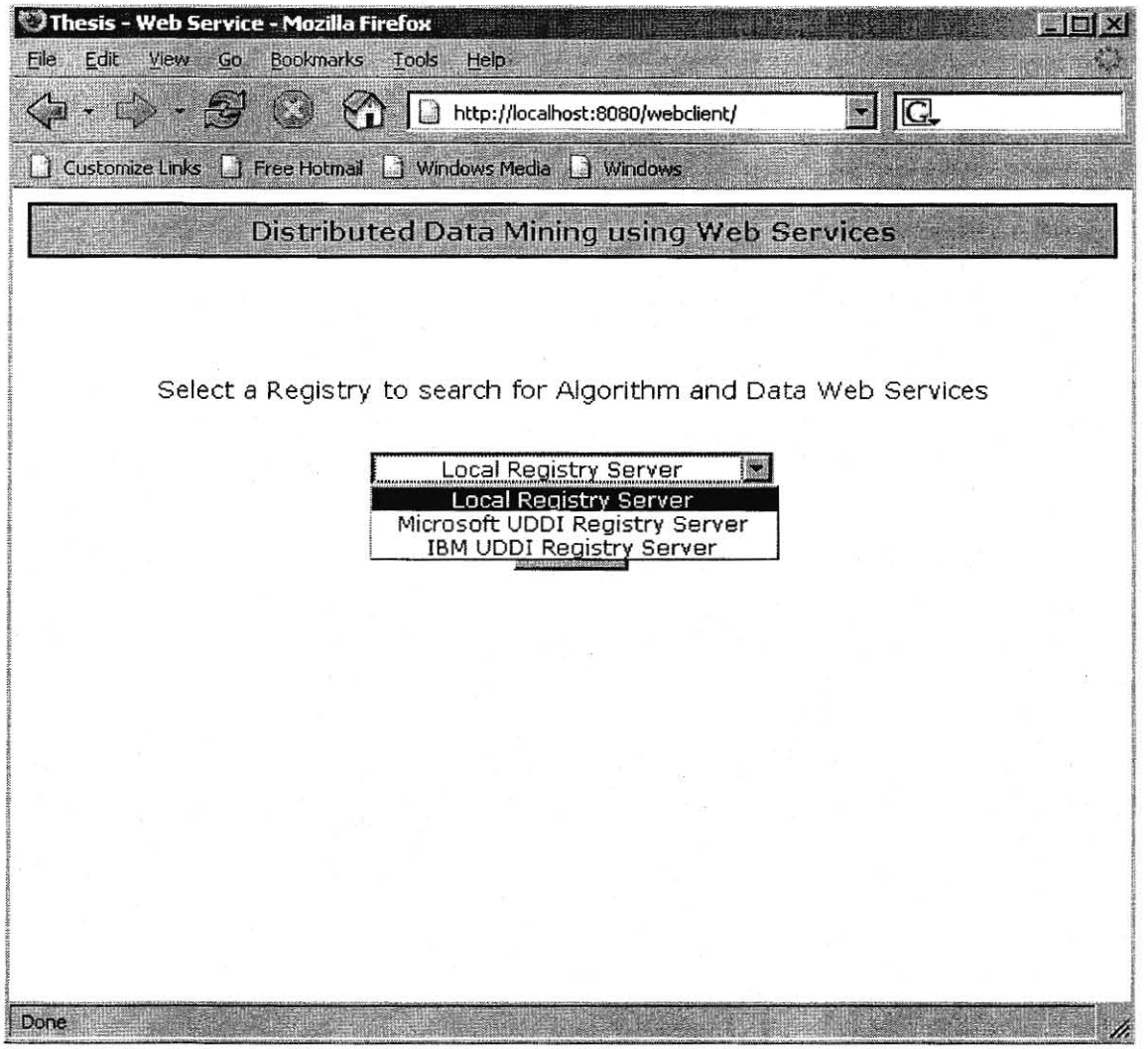

Figure 9 – Screenshot of main page showing all the Registries

Registered Web Services

Once the user picks a registry and clicks the submit button it posts the request to display_registry_services.jsp. This page queries the selected registry for Algorithm and the Data Web Services using JAXQuery Class and shows all the web services to the user. Figure 10 shows a screen shot when IBM UDDI Registry Server is picked.

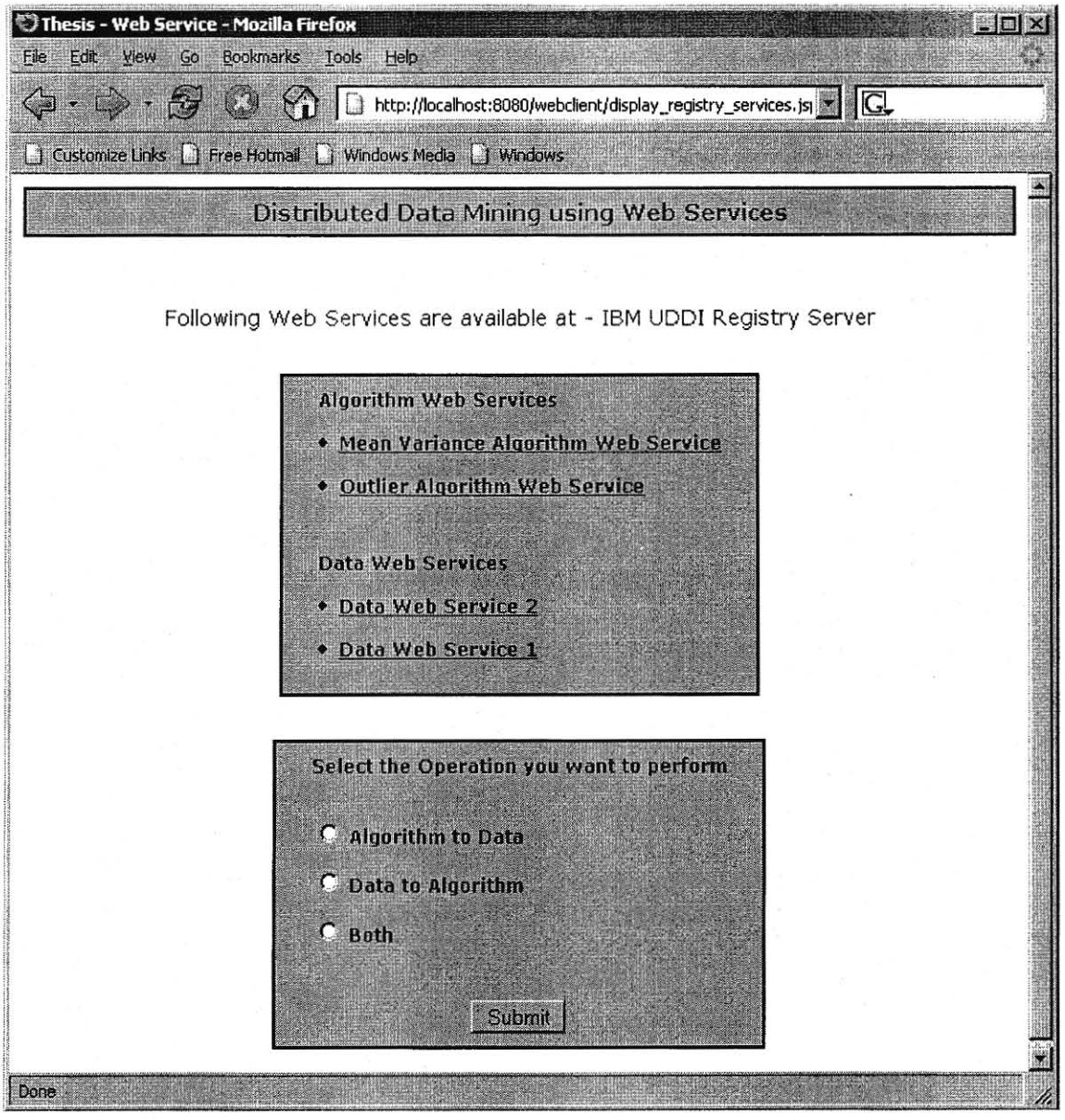

Figure 10 – Screenshot of Web Services available at Registry

The registered web services are displayed as links to allow the user to view the web service and its WSDL. Figure 11 shows a screen shot when Data Web Service 1 is clicked. Figure 12 shows WSDL of Data Web Service 1.

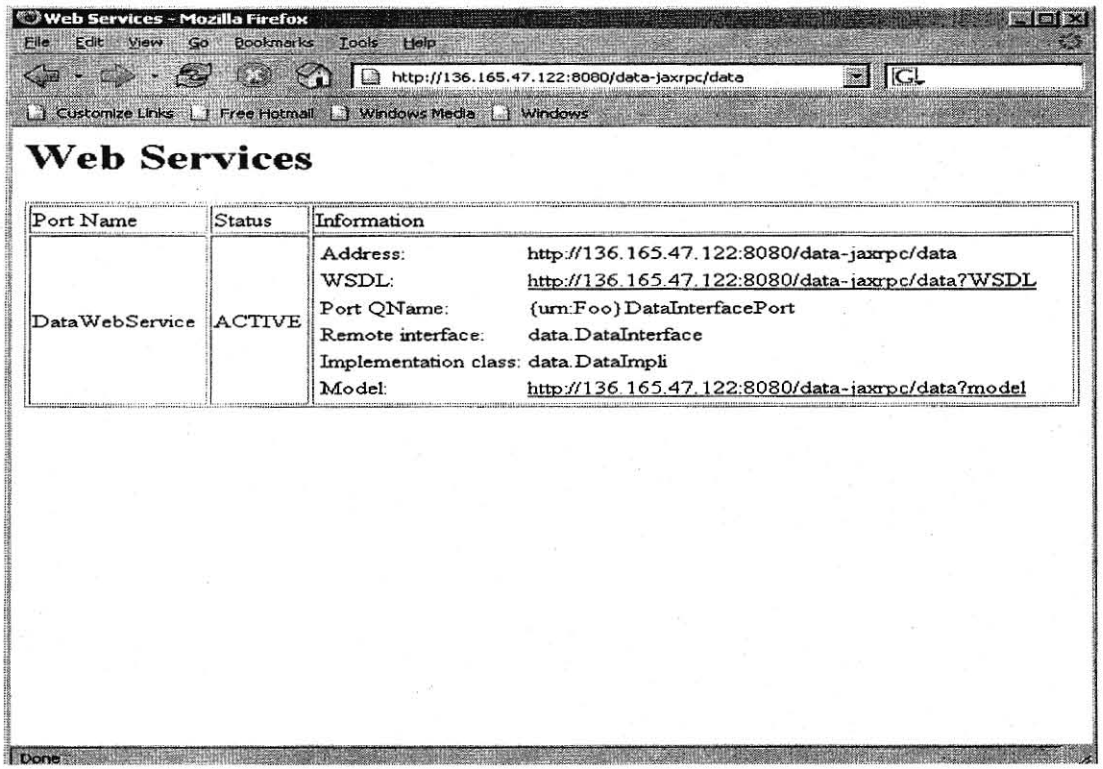

Figure 11 – Screenshot of Data 1 Web Service

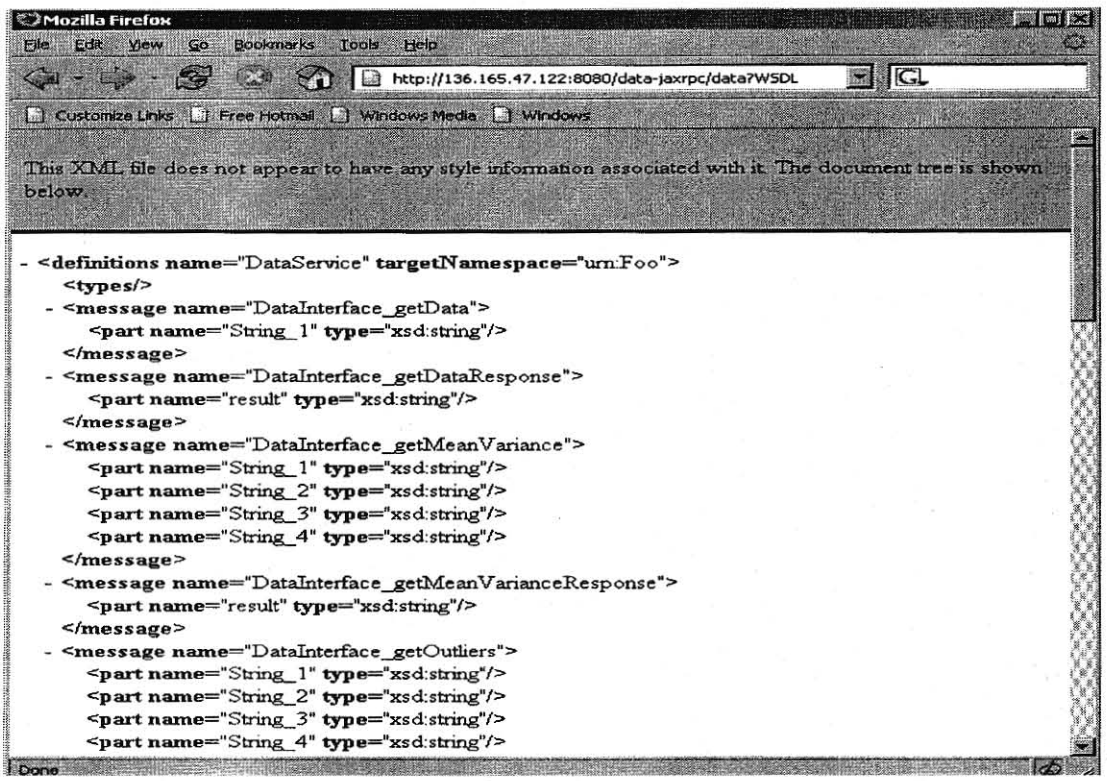

Figure 12 – Screenshot of Data 1 Web Service's WSDL file

Algorithm to Data

Web Services interaction is a major part of the framework. If the user picks the algorithm to data option in Figure 10, then Data web services call Algorithm web services to get the algorithm and run these algorithms with their data. Figure 13 shows a result of Algorithm to Data Interaction.

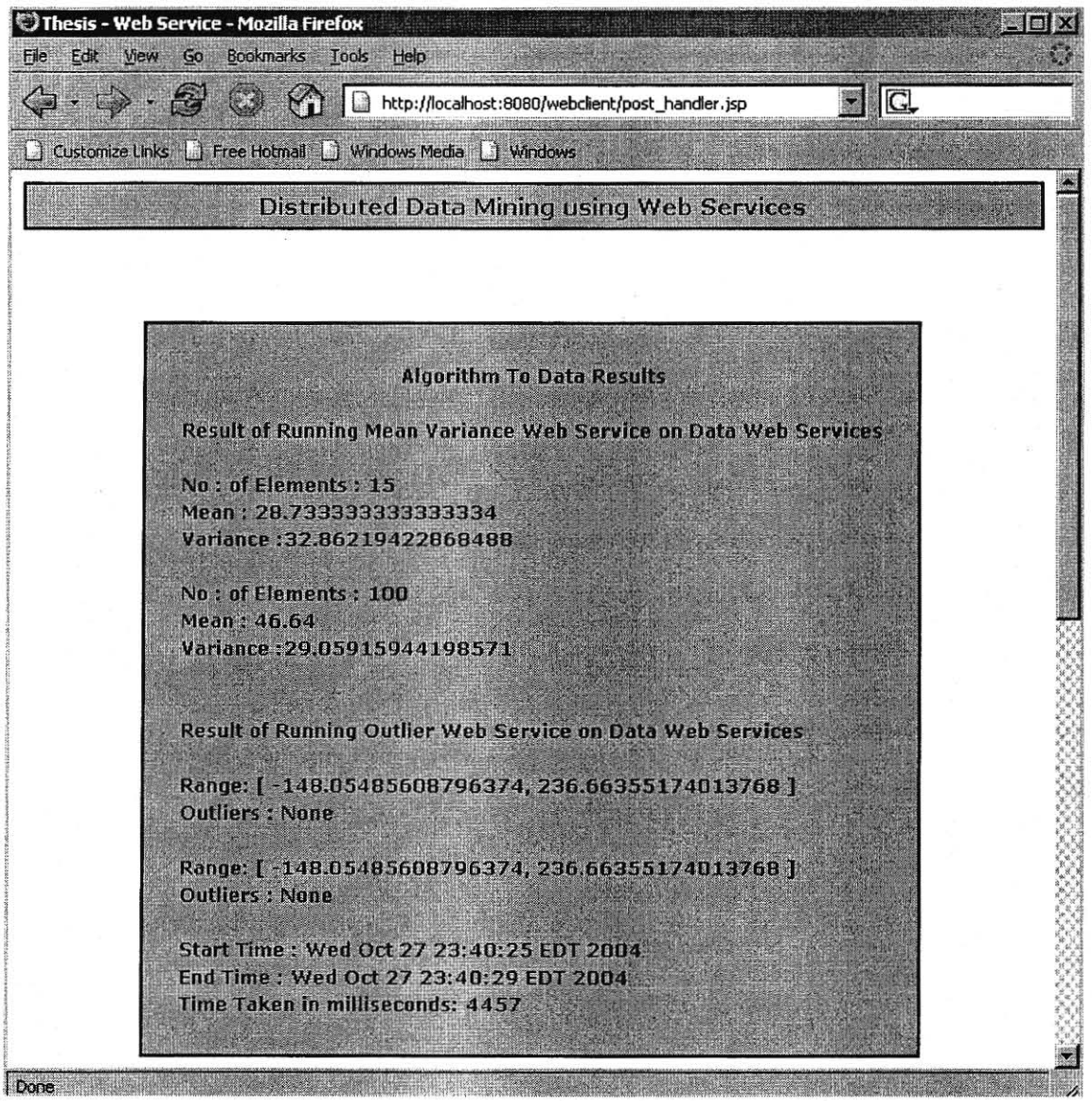

Figure 13 – Screenshot of Algorithm to Data

Data to Algorithm

This is similar to that of Algorithm to Data. If the user picks the data to algorithm option in Figure 10, then Algorithm web services call Data web services to get the data and run the algorithm with this data. Figure 14 shows a result of Data to Algorithm Interaction.

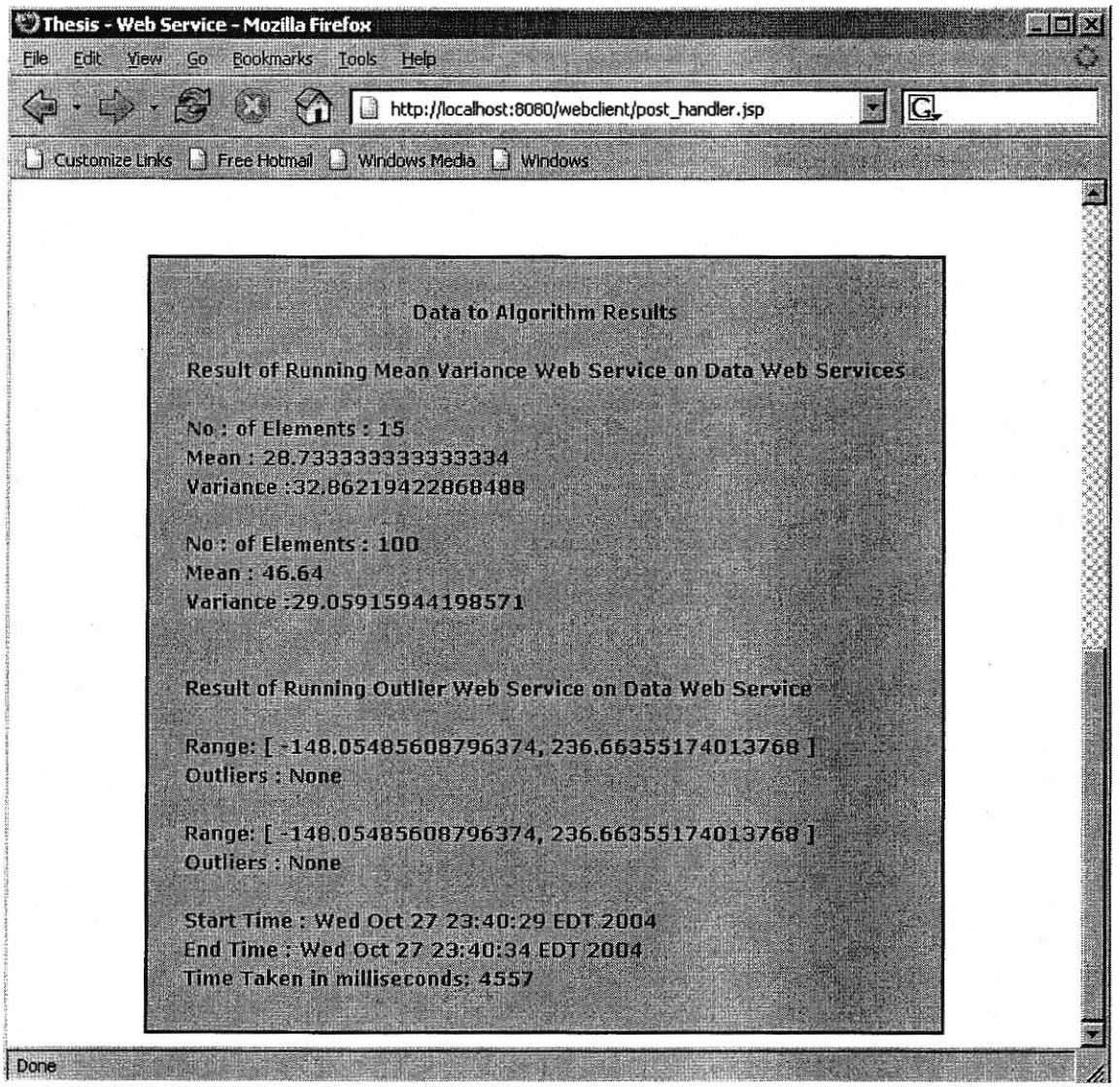

Figure 14 – Screenshot of Data to Algorithm interaction

Command Line interface

Command Line interface is similar to that of Web interface. The first thing the user sees are all the registries available and an option to pick one. Figure 15 shows all three registry servers.

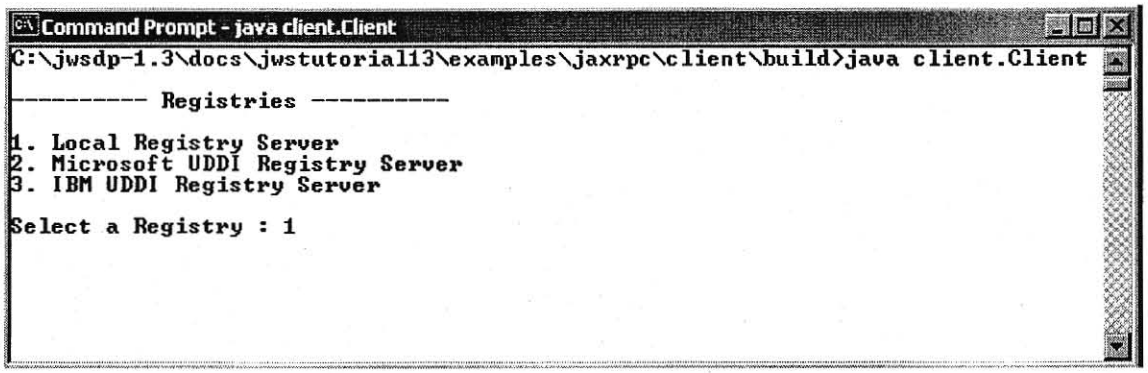

Figure 15 – Main screen of command line interface

Once the user selects a registry, all the registered Algorithm and Data web services are shown. Figure 16 shows all the Algorithm web services along with the URI and the WSDL of the service.

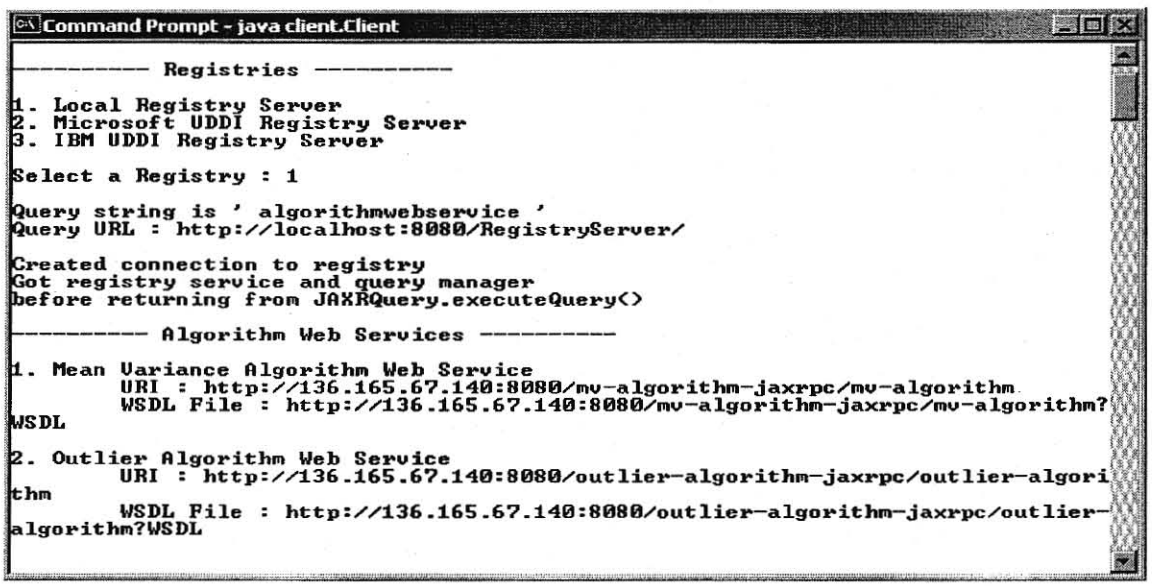

Figure 16 – Screen showing Algorithm Web Services information

Figure 17 shows the Data Web Services. The interface then presents the user with execution options.

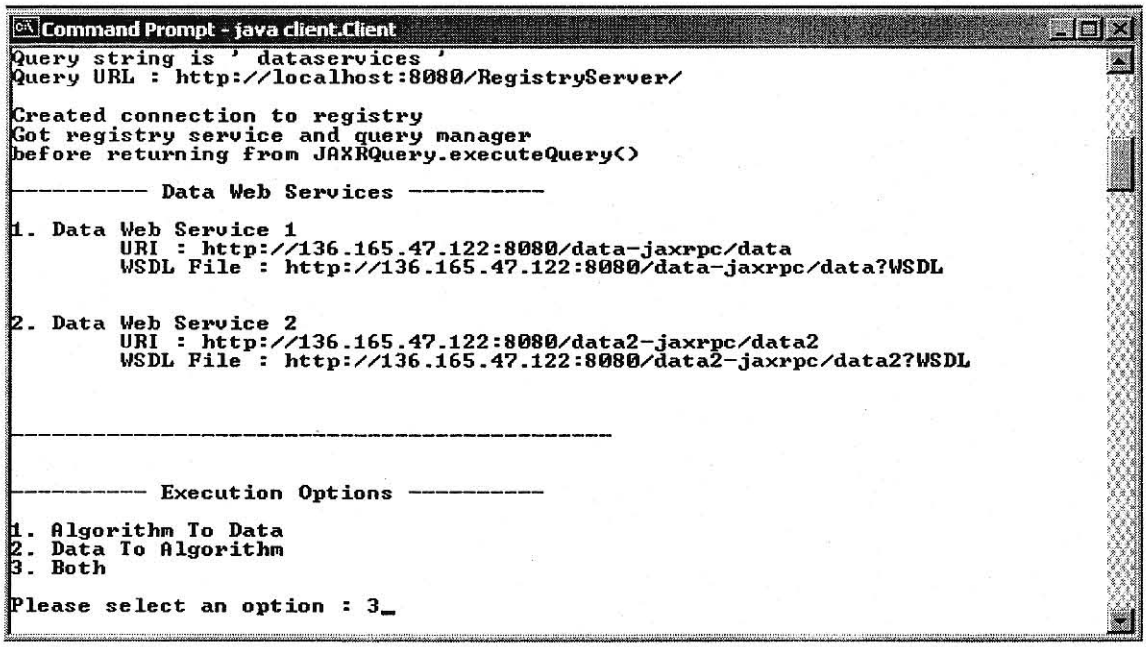

Figure 17 – Screen showing Data Web Services information

Web Services interaction involves Algorithm to Data and Data to Algorithm. Figure 18 shows Data to Algorithm interaction. In Data to Algorithm interaction, first Mean Variance web service calls Data web service 1 and Data web service 2 to get the data and then runs this data with its algorithm, then combined mean and combined variance is calculated. Finally, Outlier web service calls Data web services to return the outliers.

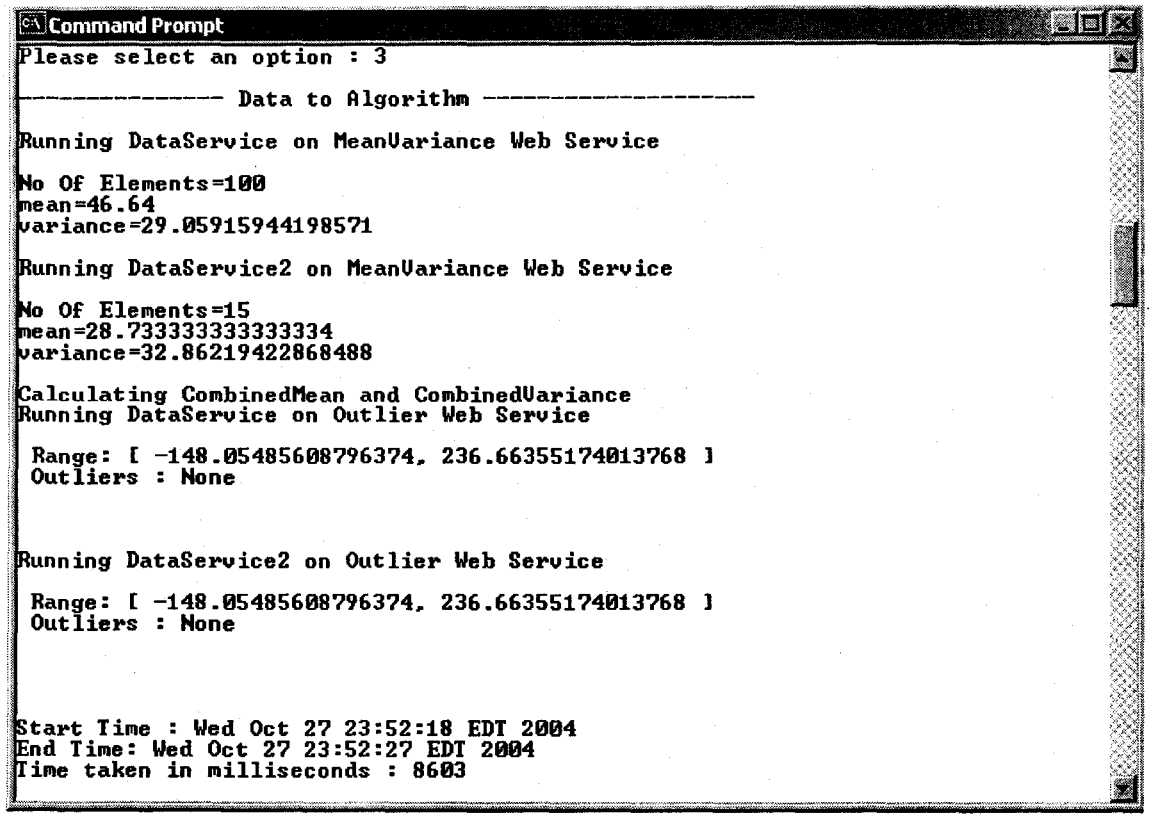

Figure 18 – Screen showing Data to Algorithm interaction

Figure 19 shows Algorithm to Data interaction. In Algorithm to Data, Data Web Services first call Mean Variance Algorithm web service to get the mean variance algorithm. This algorithm is run with their local data. Data web services then call Outlier Algorithm web service to evaluate outliers in their local data.

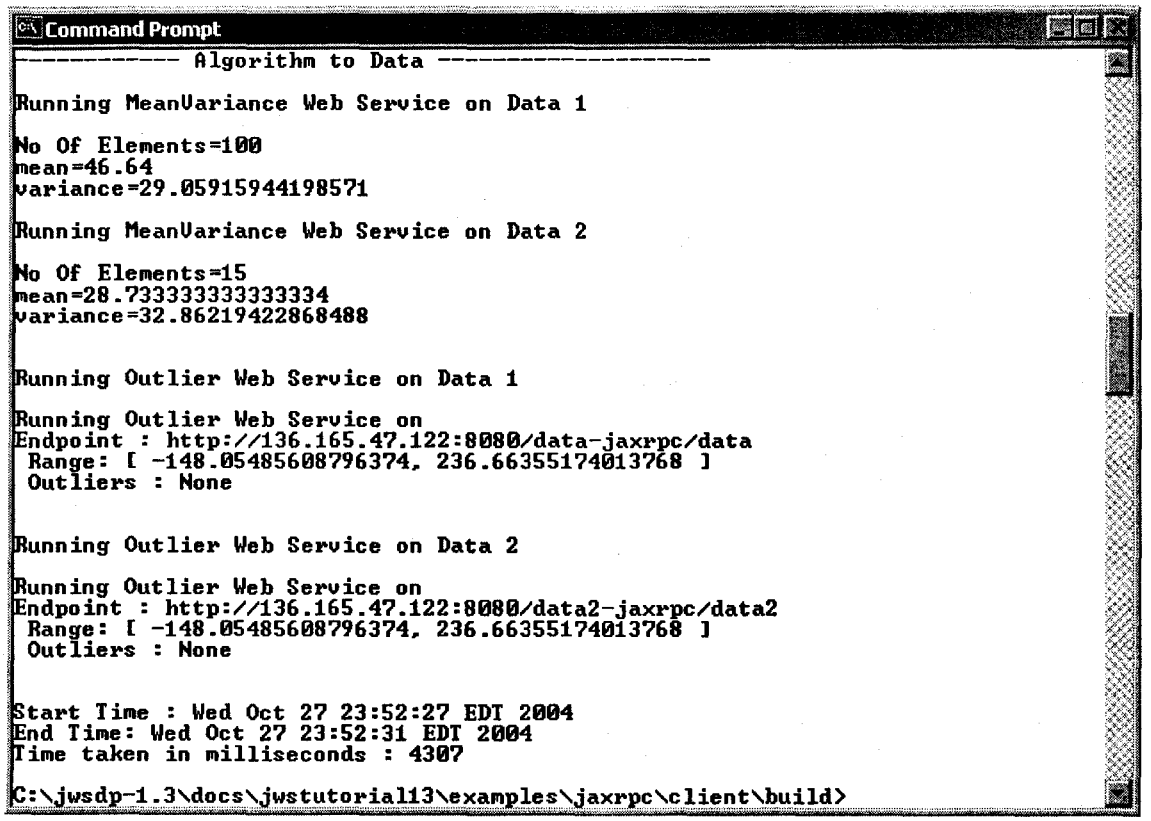

Figure 19 – Screen showing Algorithm to Data interaction

V. RESULTS AND DISCUSSION

Time taken to execute the algorithm with varying data was recorded. This was done for both the architectures with all three registry servers and a graph was plotted. Figures 20 through 25 show these graphs. X-axis represents the number of records and Y-axis, the time taken in milli seconds.

Impact of Algorithms

The time taken to execute Mean Variance Algorithm and Outlier algorithm is recorded for varying number of records. It can be seen in all the graphs that if the number of records was less, both the algorithms took about the same time, but as the number of records is increased the mean variance algorithm took more time. This can be explained as follows. For calculating mean variance, the Data web service as to iterate through the result set twice. First Data web service iterates over the result set to calculate the Mean and then using the calculated mean, it iterates over the result set second time to calculate the variance. In contrast, for calculating outliers Data web service as to iterate through the result set just once.

Impact of architectures

It can be seen from the graphs that if the number of records was less, both the architectures took about the same time. In fact Data to Algorithm took less time but as the number of records were increased the Data to Algorithm took more time. This can be easily explained. As the data is getting bigger and bigger, time taken to send this data over the network is also increasing. Also, the Data web service wraps the data in an XML format, which has to be parsed by Algorithm web service. This also adds to the delay. Clearly, the Algorithm to Data architecture is efficient and should be used in distributed data mining.

Impact of Registry Servers.

As expected the Local Registry server was the fastest among the three registry servers. There is not much difference in using IBM UDDI Registry server and Microsoft UDDI Registry server, both took about the same time.

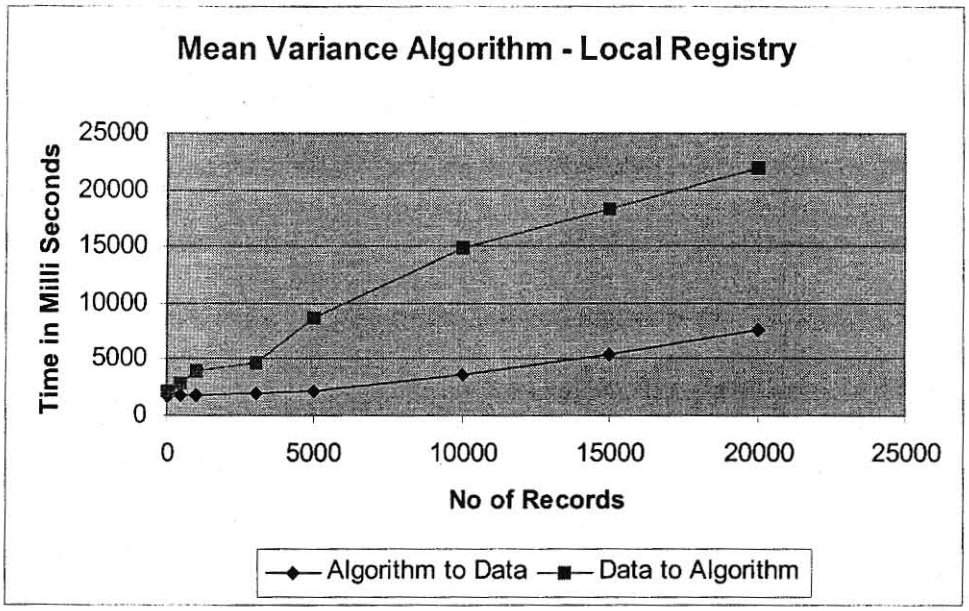

Figure 20 – Running Mean Variance Algorithm using Local Registry Server

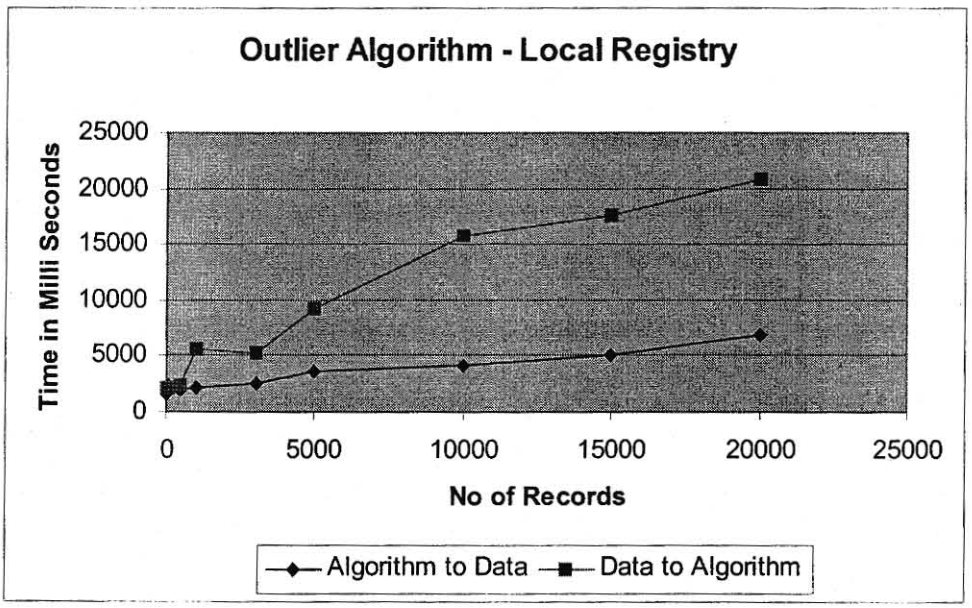

Figure 21 – Running Outlier Algorithm using Local Registry Server

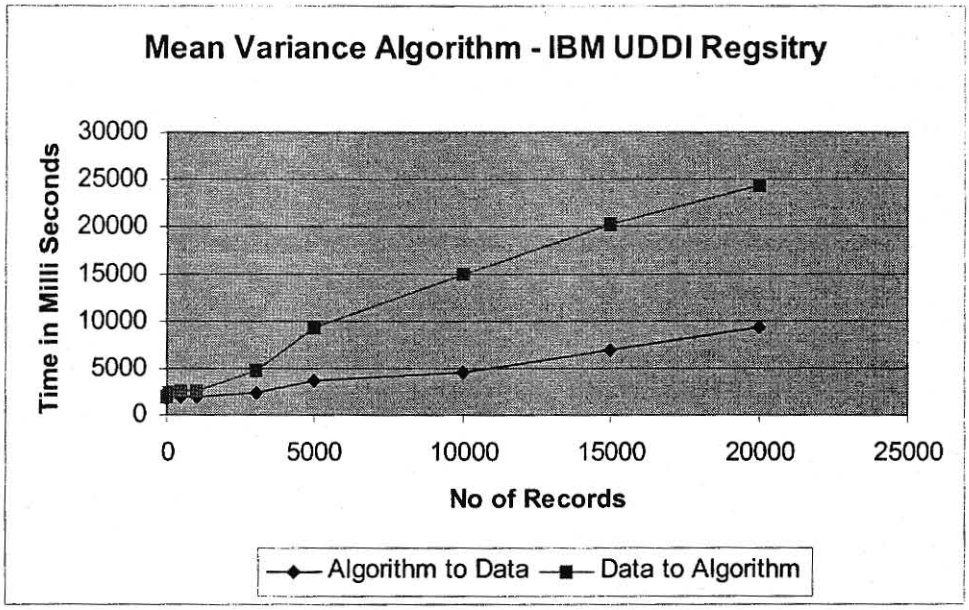

Figure 22 – Running Mean Variance Algorithm using IBM UDDI Registry Server

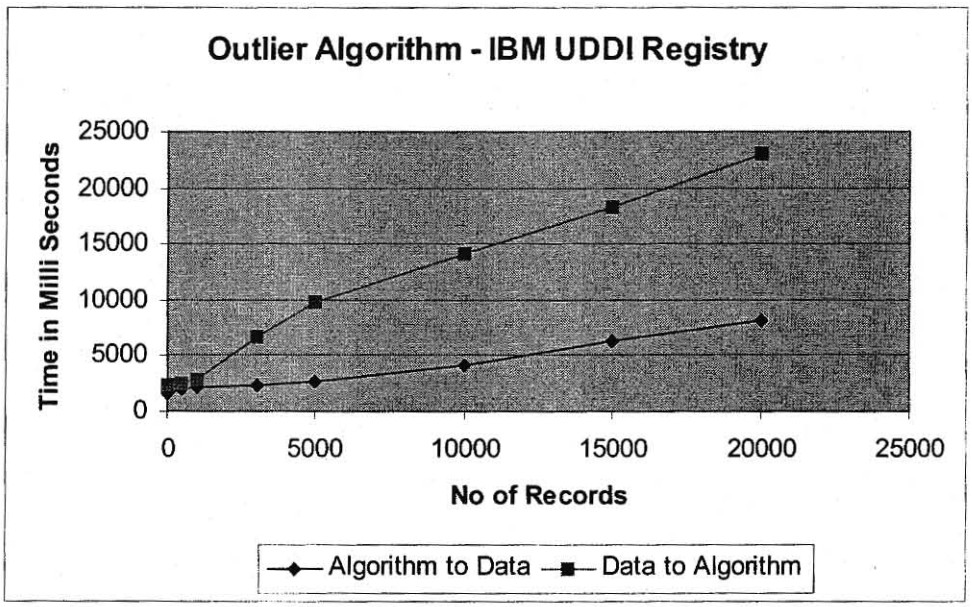

Figure 23 – Running Outlier Algorithm using IBM UDDI Registry Server

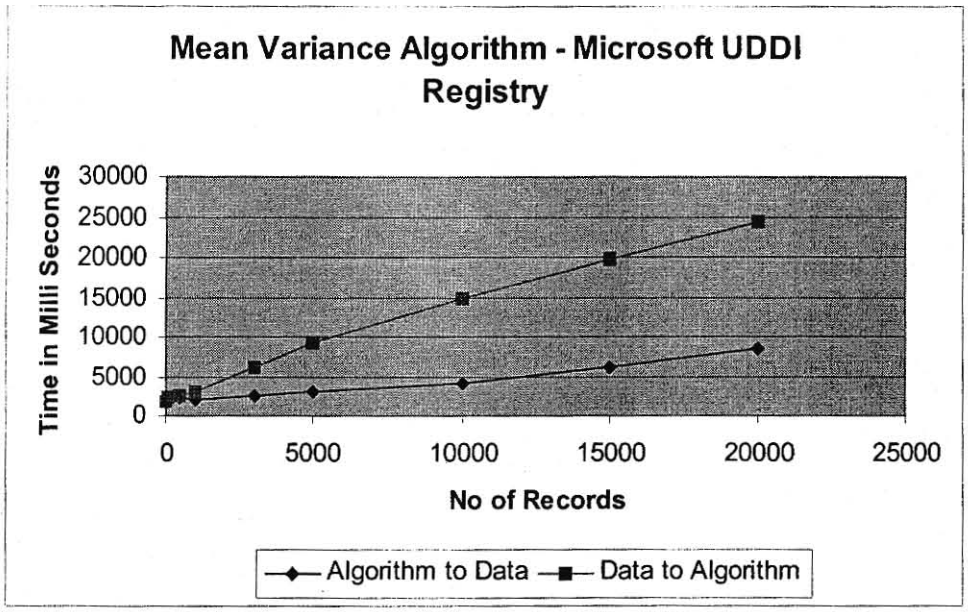

Figure 24 – Running Mean Variance Algorithm using Micrsoft UDDI Registry Server

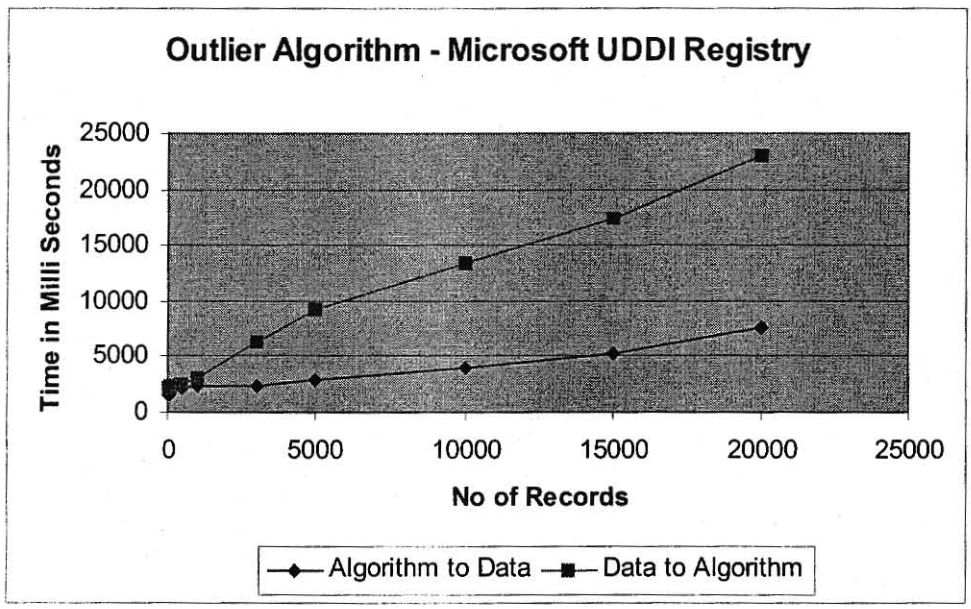

Figure 25 – Running Outlier Algorithm using Micrsoft UDDI Registry Server

VI. CONCLUSION AND RECOMMENDATIONS

The Distributed Data mining architecture using web services was implemented as planned. The result is a working model in which the mining algorithm as well as data can be moved. This was possible because the data and the algorithm were designed as web services and by registering these services at public registry servers showed that this implementation is realistic and usable framework. With web services emerging as standard we can expect more use of web services in remote and distributed data mining applications. The graphs in the last chapter have shown that Algorithm to Data architecture is efficient but when compared with the centralized data mining system both the distributed architectures have an exceptionally faster response.

Recommendations for Future Work

The main recommendation for future work is to make the system available through wireless devices. As these devices become less expensive and more widely used, this will be an important issue. Security hasn't been discussed in this thesis, adding a security layer to the system would make it reliable, robust system. Another recommendation is to allow combining of data from different data sources before running the algorithm.

REFERENCES

[1] Sun Microsystems. 2004. Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification. Avaialble from http://java.sun.com/j2se/1.4.2/docs/api/. Accessed Spring 2004, Summer 2004, Fall 2004.

[2] Subramonian, R. and Parthasarathy, S. "An Architecture for Distributed Data Mining", Fourth International Conference on Knowledge Discovery and Data Mining, New York, 1998, pp. 44-59.

[3] Kantardzic, M. and Kumar, A., "Toward Autonomic Distributed Data Mining With Intelligent Web Services", Proceedings of The 2003 International Conference on Information and Knowledge Engineering, Las Vegas, June 23-26, 2003, pp. 544-550.

[4] Sun Microsystems. 2004. Java Technology and Web Services. Available from http://java.sun.com/webservices/jwsdp/index.jsp . Accessed Spring 2004, Summer 2004, Fall 2004.

[5] Sun Microsystems. 2004. Java API for XML-Based RPC (JAX-RPC). Available from http://java.sun.com/xml/jaxrpc/overview.html. Accessed Spring 2004, Summer 2004, Fall 2004.

[6] Sun Microsystems. 2004. Java API for XML Registries (JAXR). Available from http://java.sun.com/xml/jaxr/overview.html. Accessed Spring 2004, Summer 2004, Fall 2004.

[7] Sun Microsystems. 2004. Java Servlet Technology. Available from http://java.sun.com/products/servlet/overview.html. Accessed Spring 2004, Summer 2004, Fall 2004.

[8] Sun Microsystems. 2004. JavaServer Pages Standard Tag Library Available from http://java.sun.com/products/jsp/jstl/index.jsp. Accessed Spring 2004, Summer 2004, Fall 2004.

[9] JDOM 2004. Available from http://www.jdom.org. Accessed Spring 2004, Summer 2004, Fall 2004.

[10] Sun Microsystems. 2004. The Java Web Services Tutorial. Available from http://java.sun.com/webservices/docs/1.3/tutorial/doc/index.html. Accessed Spring 2004, Summer 2004, Fall 2004.

[11] Professional Java Web Services. 2002. Wrox Press.

[12] Marty Hall. 2002. More Servlets and JavaServer Pages. Prentice Hall.

APPENDIX – A CODE LISTINGS

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: MVAlgorithmInter.java
* Purpose: Mean Variance Algorithm Interface
*/


package meanvariancealgorithm;

import javax.activation.DataHandler;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MVAlgorithmInter extends Remote {

    public DataHandler getAlgorithmFile(String name)
            throws RemoteException;

    public String getDataRunMeanVariance(String qNameService,String qNamePort,
                                         String endPoint,String nameSpace )
            throws RemoteException;
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: MVAlgorithmInter.java
* Purpose: Mean Variance Algorithm Implementation Class
*/

package meanvariancealgorithm;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

import javax.activation.DataHandler;
import javax.xml.namespace.QName;
import javax.xml.rpc.*;
import java.io.*;
import java.rmi.RemoteException;
import java.util.List;

public class MVAlgorithmImpli implements MVAlgorithmInter {

    private static String CLASSNAME="meanvariancealgorithm.MVAlgorithmImpli" ;
     private static String ENCODING_STYLE_PROPERTY
                          ="javax.xml.rpc.encodingstyle.namespace.uri";
     private static String NS_XSD = "http://www.w3.org/2001/XMLSchema";
     private static String URI_ENCODING =
                     "http://schemas.xmlsoap.org/soap/encoding/";

     private static String dirLocation="c:/jwsdp-1.3/docs/ jwstutorial13/
                      examples/jaxrpc/meanvariancealgorithm/algorithms/";
     private static String compilerLocation = "c:/j2sdk1.4.1_01/bin/javac";
     private static String javaLocation = "c:/j2sdk1.4.1_01/bin/java";

     /**
      * Returns the file requested as a DataHandler Object
      * @param name - name of the java file
      * @return DataHandler object
      */

     public DataHandler getAlgorithmFile(String name){
         String fileDir = "c:/jwsdp-1.3/docs/jwstutorial13/examples/jaxrpc/"+
                          "meanvariancealgorithm/algorithms/";
         String methodName = ".getAlgorithmFile()";
         String temp=null;
         String stringOutput =new String();
         StringBuffer stringBuffer = new StringBuffer();

         BufferedReader bufferedReader;
         DataHandler dataHandler = null;

         System.out.println("Entered " + CLASSNAME + methodName);
         System.out.println("File Dir : " + fileDir);
         System.out.println("File Requested : " + name);

         try{
              File file = new File(fileDir+name);
               bufferedReader = new BufferedReader(new FileReader(file));
```

```
            temp=bufferedReader.readLine();

            while(temp != null){
                  stringBuffer.append(temp);
                  temp=bufferedReader.readLine();
                  stringBuffer.append("\n");
            }
           stringOutput = stringBuffer.toString();
           dataHandler  = new DataHandler(stringOutput, "text/plain");

       } catch (Exception ex){
            System.out.println(CLASSNAME + methodName +" caught Exception : "
                               + ex.toString());
       }
       return dataHandler;
}


/**
 * Gets the Data from Data web service and runs its algorithm
 * and returns the result as String
 * @param qNameService - Data Web Service name
 * @param qNamePort - Data Web Service port
 * @param endPoint - Data Web Service end point
 * @param nameSpace - Data Web Service name space
 * @return result as String
 */

public String getDataRunMeanVariance(String qNameService,String qNamePort,
                                     String endPoint, String nameSpace){

    String methodName=".getDataRunMeanVariance()";
    System.out.println("Entered " + CLASSNAME + methodName);

    String output="";
    Call call = prepareCallObject(qNameService, qNamePort
                                  endPoint,nameSpace);
    String params[] ={"T4"};

    String javaFileName = "MeanVariance.java";
    String classFileName= "MeanVariance";

    try {
        output =(String) call.invoke(params);

        String data = parseXMLData(output);

         String executionCommand = javaLocation + " -cp " +
                       dirLocation + " "+ classFileName + " "+data;

        output = execute(javaFileName, executionCommand);

        System.out.println("output : " + output);

    } catch (RemoteException e) {
        System.out.println(CLASSNAME + methodName + " caught
                           RemoteException : " + e.toString());
    }

    return output;
}
```

```
/**
 * Prepares a Call object to call the Data web service
 * @param qNameService - Data Web Service name
 * @param qNamePort - Data Web Service port
 * @param endPoint - Data Web Service end point
 * @param nameSpace - Data Web Service name space
 * @return Call object
 */
private Call prepareCallObject(String qNameService, String qNamePort,
                               String endPoint, String nameSpace){
    Call call = null;
    String methodName = ".prepareCallObject()";
    System.out.println("Entered " + CLASSNAME + methodName);

    try {
        ServiceFactory factory = ServiceFactory.newInstance();
        Service service = factory.createService(new QName(qNameService));

        QName port = new QName(qNamePort);

        call = service.createCall(port);
        call.setTargetEndpointAddress(endPoint);

        call.setProperty(Call.SOAPACTION_USE_PROPERTY, new Boolean(true));
        call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
        call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

        QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
        call.setReturnType(QNAME_TYPE_STRING);
        call.setOperationName(new QName(nameSpace, "getData"));
        call.addParameter("String_1", QNAME_TYPE_STRING, ParameterMode.IN);

    } catch (ServiceException e) {
        System.out.println(CLASSNAME + methodName + " caught
                           ServiceException : " + e.toString());
    }
    return call;
}


/**
 * Parses the XML to get the actual data
 * @param xml - XML String
 * @return data as comma separated String
 */

public String parseXMLData(String xml){
    String methodName = ".parseXMLData()";
    System.out.println("Entered " + CLASSNAME + methodName);

    StringBuffer stringBuffer = new StringBuffer();
    try {
        SAXBuilder builder = new SAXBuilder();
        Document doc =  builder.build(new StringReader(xml));
        Element root = doc.getRootElement();
        List childerList = root.getChildren();
        for (int i = 0; i < childerList.size(); i++) {
            Element row = (Element) childerList.get(i);

            List dataList = row.getChildren();
            for (int j = 0; j < dataList.size(); j++) {
                Element data= (Element) dataList.get(j);
                stringBuffer.append(data.getText());
```

```
                    stringBuffer.append(",");
                }
            }
        } catch (JDOMException e) {
            System.out.println(CLASSNAME + methodName+ " caught
                                JDOMException : " +e.toString());
        }
        return stringBuffer.toString();
    }


    /**
     * Compiles and executes the java file and returns the result
     * as String
     * @param javaFileName - java file name
     * @param executionCommand - command to execute the class file
     * @return result of executing the class file with the data String
     */

    private String execute(String javaFileName,
                           String executionCommand){
        String temp = "";
        String methodName =".executeFile()";
        String compileCommand = compilerLocation + " " +
                                dirLocation + javaFileName;

        System.out.println("Entered "+ CLASSNAME + methodName);
        System.out.println("Compile Command : " + compileCommand);
        System.out.println("Execution Command : " + executionCommand);
        try {
            Runtime runtime = Runtime.getRuntime();
            Process process = runtime.exec(compileCommand);

            while(!( new File(dirLocation+"MeanVariance.class")).exists()){
                System.out.println("MeanVariance Class File doesnt exits");
                Thread.sleep(1000);
            }

            process = runtime.exec(executionCommand);

            BufferedReader bufferedReader = new BufferedReader(new
                    InputStreamReader(process.getInputStream()));

            String out = "";
            StringBuffer sb = new StringBuffer();
            b.append(bufferedReader.readLine());
            sb.append("\n");
            while ((out = bufferedReader.readLine()) != null) {
                sb.append(out);
                sb.append("\n");
            }
            temp = sb.toString();
        } catch (IOException e) {
            System.out.println(CLASSNAME + methodName + " caught
                                IOException : " +e.toString());
        } catch (InterruptedException e) {
            System.out.println(CLASSNAME + methodName + " caught
                                InterruptedException : " +e.toString());
        }
       return temp;
    }
}
```

OutlierAlgorithmInter.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: OutlierAlgorithmInter.java
* Purpose: Outlier Algorithm Implementation Class
*/

package outlieralgorithm;

import javax.activation.DataHandler;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface OutlierAlgorithmInter extends Remote {
    public DataHandler getAlgorithmFile(String name)
       throws RemoteException;

    public String getDataRunOutliers(String qNameService,String qNamePort,
                                      String endPoint,String nameSpace,
                                      String combinedMean,
                                      String combinedVariance)
       throws RemoteException;
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                        Fall 2004
*
* File: OutlierAlgorithmImpli.java
* Purpose: Outlier Algorithm Implementation Class
*/

package outlieralgorithm;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

import javax.activation.DataHandler;
import javax.xml.namespace.QName;
import javax.xml.rpc.*;
import java.io.*;
import java.rmi.RemoteException;
import java.util.List;

public class OutlierAlgorithmImpli implements OutlierAlgorithmInter{

    private static String CLASSNAME = "outlieralgorithm.OutlierAlgorithmImpli";
    private static String ENCODING_STYLE_PROPERTY
                                  ="javax.xml.rpc.encodingstyle.namespace.uri";
    private static String NS_XSD  = "http://www.w3.org/2001/XMLSchema";
    private static String URI_ENCODING
                           = "http://schemas.xmlsoap.org/soap/encoding/";

    private static String dirLocation="c:/jwsdp-1.3/docs/jwstutorial13/
                         examples/ jaxrpc/outlieralgorithm/algorithms/";
    private static String compilerLocation = "c:/j2sdk1.4.1_01/bin/javac";
    private static String javaLocation = "c:/j2sdk1.4.1_01/bin/java";


    /**
     * Returns the file requested as a DataHandler Object
     * @param name - name of the java file
     * @return DataHandler object
    */

    public DataHandler getAlgorithmFile(String name){
        String fileDir = "c:/jwsdp-1.3/docs/jwstutorial13/examples/jaxrpc/
                          outlieralgorithm/algorithms/";
        String methodName=".getAlgorithmFile()";
        String temp=null;
        String stringOutput =new String();
        StringBuffer stringBuffer = new StringBuffer();

        BufferedReader bufferedReader;
        DataHandler dataHandler = null;

        System.out.println("Entered " + CLASSNAME + methodName);
        System.out.println("File Dir : " + fileDir);
        System.out.println("File Requested : " + name);

        try{
            File file = new File(fileDir+name);
```

```
            bufferedReader = new BufferedReader(new FileReader(file));
            temp=bufferedReader.readLine();

            while(temp != null){
                  stringBuffer.append(temp);
                  temp=bufferedReader.readLine();
                  stringBuffer.append("\n");
            }
             stringOutput = stringBuffer.toString();
             dataHandler  = new DataHandler(stringOutput , "text/plain");

      } catch (Exception ex){
            System.out.println(CLASSNAME + methodName + " caught Exception : "
                              + ex.toString());
      }
      return dataHandler;
}


/**
 * Gets the Data from Data web service and runs its algorithm and
 * returns the result as String
 * @param qNameService - Data Web Service name
 * @param qNamePort - Data Web Service port
 * @param endPoint - Data Web Service end point
 * @param nameSpace - Data Web Service name space
 * @return result as String
 */

public String getDataRunOutliers(String qNameService, String qNamePort,
                                 String endPoint, String nameSpace,
                                String combinedMean, String combinedVariance){

      String methodName=".getDataRunOutliers()";
      System.out.println("Entered " + CLASSNAME + methodName);

      String output="";
      Call call = prepareCallObject(qNameService, qNamePort,
                                    endPoint, nameSpace);
      String params[] ={"T4"};

      String javaFileName = "Outliers.java";
      String classFileName= "Outliers";

      try {
          output =(String) call.invoke(params);

          String data = parseXMLData(output);

          String executionCommand = javaLocation + " -cp " + dirLocation + "
                                    "+ classFileName + " "+data+" " +
                                    combinedMean + " " + combinedVariance;

          output = execute(javaFileName, executionCommand);

          System.out.println("output : " + output);

      } catch (RemoteException e) {
          System.out.println(CLASSNAME + methodName + " caught
                             RemoteException : " + e.toString());
      }
      return output;
}
```

```
/**
 * Prepares a Call object to call the Data web service
 * @param qNameService - Data Web Service name
 * @param qNamePort - Data Web Service port
 * @param endPoint - Data Web Service end point
 * @param nameSpace - Data Web Service name space
 * @return Call object
 */

private Call prepareCallObject(String qNameService, String qNamePort,
                               String endPoint, String nameSpace){
    Call call = null;

    String methodName = ".prepareCallObject()";
    System.out.println("Entered " + CLASSNAME + methodName);

    try {
        ServiceFactory factory = ServiceFactory.newInstance();
        Service service = factory.createService(new QName(qNameService));

        QName port = new QName(qNamePort);

        call = service.createCall(port);
        call.setTargetEndpointAddress(endPoint);

        call.setProperty(Call.SOAPACTION_USE_PROPERTY, new Boolean(true));
        call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
        call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

        QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
        call.setReturnType(QNAME_TYPE_STRING);
        call.setOperationName(new QName(nameSpace, "getData"));
        call.addParameter("String_1", QNAME_TYPE_STRING, ParameterMode.IN);

    } catch (ServiceException e) {
        System.out.println(CLASSNAME + methodName + " caught
                           ServiceException : " + e.toString());
    }

    return call;
}


/**
 * Parses the XML to get the actual data
 * @param xml - XML String
 * @return data as comma separated String
 */

public String parseXMLData(String xml){

    String methodName = ".parseXMLData()";
    System.out.println("Entered " + CLASSNAME + methodName);

    StringBuffer stringBuffer = new StringBuffer();
    try {
        SAXBuilder builder = new SAXBuilder();
        Document doc =  builder.build(new StringReader(xml));
        Element root = doc.getRootElement();
        List childerList = root.getChildren();
        for (int i = 0; i < childerList.size(); i++) {
            Element row = (Element) childerList.get(i);
```

```
                List dataList = row.getChildren();
                for (int j = 0; j < dataList.size(); j++) {
                    Element data= (Element) dataList.get(j);
                    stringBuffer.append(data.getText());
                    stringBuffer.append(",");
                }
            }
        } catch (JDOMException e) {
            System.out.println(CLASSNAME + methodName+ " caught JDOMException :
                               " +e.toString());
        }
        return stringBuffer.toString();
    }


    /**
     * Compiles and executes the java file and returns the result as String
     * @param javaFileName - java file name
     * @param executionCommand - command to execute the class file
     * @return result of executing the class file with the data as String
     */

    private String execute(String javaFileName, String executionCommand){
        String temp = "";
        String methodName =".executeFile()";
        String compileCommand = compilerLocation + " " +
                                dirLocation + javaFileName;

        System.out.println("Entered "+ CLASSNAME + methodName);
        System.out.println("Compile Command : " + compileCommand);
        System.out.println("Execution Command : " + executionCommand);
        try {
            Runtime runtime = Runtime.getRuntime();
            Process process = runtime.exec(compileCommand);

            while(! (new File(dirLocation+"Outliers.class")).exists()){
                System.out.println("Outliers Class File doesnt exits");
                Thread.sleep(1000);
            }

            process = runtime.exec(executionCommand);
            BufferedReader bufferedReader = new BufferedReader(new
                    InputStreamReader(process.getInputStream()));
            String out = "";
            StringBuffer sb = new StringBuffer();
           sb.append(bufferedReader.readLine());
            sb.append("\n");
            while ((out = bufferedReader.readLine()) != null) {
                sb.append(out);
                sb.append("\n");
            }
            temp = sb.toString();
        } catch (IOException e) {
            System.out.println(CLASSNAME + methodName + " caught IOException :
                               " +e.toString());
        } catch (InterruptedException e) {
            System.out.println(CLASSNAME + methodName + " caught
                               InterruptedException : " +e.toString());
        }
      return temp;
    }
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: MeanVariance.java
* Purpose: Mean Variance Algorithm Implementation Class
*/

import java.util.StringTokenizer;

public class MeanVariance {

    private String getMeanVariance(double d[]){
        double sum=0,inter=0,summation=0;
        double mean=0, var=0, varsqr=0;

        StringBuffer result=new StringBuffer();
        result.append(d.length).append(",");
        for (int i = 0; i < d.length; i++) {
            sum+=d[i];
        }

        mean = sum/d.length;
        result.append(mean).append(",");

        for (int i = 0; i < d.length; i++) {
            inter = ((d[i] - mean) * (d[i] - mean));
            summation+=inter;
        }

        varsqr = summation / (d.length - 1);
             var = Math.sqrt(varsqr);
        result.append(var);

        return result.toString();

    }

    public static void main(String[] args) {
        MeanVariance mv = new MeanVariance();

        StringTokenizer tokenizer = new StringTokenizer(args[0], ",");
        double data[] = new double[tokenizer.countTokens()];

        int i=0;
        while(tokenizer.hasMoreTokens()){
            data[i++] = Double.parseDouble(tokenizer.nextToken());
        }
        System.out.println(mv.getMeanVariance(data));
    }
}
```

Outliers.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: Outliers.java
* Purpose: Outliers Class used to calculate the outliers for a given data set
*/

import java.util.StringTokenizer;

public class Outliers{

    private String findOutliers(double data[],double combinedMean,
                                double combinedVariance){
        double d=0;
        double r1 = combinedMean - (2 * combinedVariance);
        double r2 = combinedMean + (2 * combinedVariance);

        StringBuffer dataSet = new StringBuffer();
        dataSet.append("Data Set : [");
        StringBuffer outlier = new StringBuffer("Outliers : ");
        StringBuffer returnValue = new StringBuffer();

        for (int i = 0; i <data.length; i++) {
              d = data[i];
              dataSet.append(" " + d);
              if( (d < r1) || (d > r2))
              {
                    outlier.append(d);
                    outlier.append(" ");
              }
        }
        dataSet.append(" ]");

        if(outlier.length() == 11){
             outlier.append("None");
        }
        returnValue.append(dataSet.toString()).append("|").append("Range: [ " +
                          r1 + ", " + r2 + " ] | ").append(outlier.toString());
        return returnValue.toString();
    }

    public static void main(String args[])     {
       Outliers outliers = new Outliers();

       StringTokenizer tokenizer = new StringTokenizer(args[0], ",");
       double data[] = new double[tokenizer.countTokens()];

       int i=0;
       while(tokenizer.hasMoreTokens()){
             data[i++] = Double.parseDouble(tokenizer.nextToken());
       }
       double dm = Double.parseDouble(args[1]);
       double dv = Double.parseDouble(args[2]);

       String temp = outliers.findOutliers(data, dm,dv);
       System.out.println(temp);

       }
}
```

DataInterface.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: DataInterface.java
* Purpose: Data 1 and Data 2 Web Service Interface
*/

package data;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface DataInterface extends Remote {
    public String getMeanVariance(String endPoint,
                                  String qNameService,
                                  String qNamePort,
                                  String nameSpace)
            throws RemoteException;

    public String getOutliers(String endPoint,
                              String qNameService,
                              String qNamePort,
                              String nameSpace,
                              String combinedMean,
                              String combinedVariance)
            throws RemoteException;

    public String getData(String table) throws RemoteException;
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: DataImpli.java
* Purpose: Data 1 and Data 2 Web Service Implementation Class
*/

package data;

import java.io.*;
import javax.xml.namespace.QName;
import javax.xml.rpc.*;
import javax.activation.DataHandler;
import java.util.*;
import java.rmi.RemoteException;
import java.sql.*;
import java.net.URL;

import org.jdom.Element;
import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

public class DataImpli implements DataInterface{

    private static String ENCODING_STYLE_PROPERTY
                              ="javax.xml.rpc.encodingstyle.namespace.uri";
    private static String NS_XSD = "http://www.w3.org/2001/XMLSchema";
    private static String URI_ENCODING =
                               "http://schemas.xmlsoap.org/soap/encoding/";

    private static String CLASSNAME = "data.DataImpli";
    private static String dirLocation="c:/jwsdp-1.3/docs/jwstutorial13/
                                       examples/jaxrpc/data/";
    private static String compilerLocation = "c:/j2sdk1.4.2_04/bin/javac";
    private static String javaLocation = "c:/j2sdk1.4.2_04/bin/java";


    /**
     *
     * @param endPoint - end point of Algorithm web service
     * @param qNameService - service name of Algorithm web service
     * @param qNamePort - port of Algorithm web service
     * @param nameSpace - name space of Algorithm web service
     * @return mean and variance of the data set as String
     */

    public String getMeanVariance(String endPoint,
                                  String qNameService,
                                  String qNamePort,
                                  String nameSpace){
        String returnValue = "";
        String methodName = ".getMeanVariance()";

        System.out.println("Entered " + CLASSNAME + methodName);
        System.out.println("End Point : " +  endPoint);

        String javaFileName = "MeanVariance.java";
```

```
    String classFileName = "MeanVariance";

    try {
        // prepareCallObject prepares the Call Object with reqd parameters
        // for calling Algorithm Web Service

        Call call = prepareCallObject(qNameService, qNamePort,
                                      endPoint, nameSpace);
        String[] params = { javaFileName};
        String output=(String)  call.invoke(params);

        // createFile method creates the file in the dirLocation
        createFile(output, javaFileName);
        String data = getDataAsString("T4");

        String executionCommand = javaLocation + " -cp " + dirLocation + "
                                  "+ classFileName + " "+data;
        // executeFile method executes the file and returns the output
        returnValue = executeFile(javaFileName,classFileName,
                                  executionCommand);
    } catch (Exception e) {
             System.out.println(CLASSNAME + methodName + " caught Exception
                                : " + e.toString());
    }
    return returnValue;
}


/**
 *
 * @param endPoint - end point of Algorithm web service
 * @param qNameService - service name of Algorithm web service
 * @param qNamePort - port of Algorithm web service
 * @param nameSpace - name space of Algorithm web service
 * @param combinedMean - combinedMean from 2 data sets
 * @param combinedVariance - combined variacne from 2 data sets
 * @return outliers, range and data set as String object
 */

public String getOutliers(String endPoint,
                          String qNameService,
                          String qNamePort,
                          String nameSpace,
                          String combinedMean,
                          String combinedVariance){
    String returnValue = "";
    String methodName = ".getOutliers()";

    System.out.println("Entered " + CLASSNAME + methodName);
    System.out.println("End Point : " + endPoint);

    String javaFileName = "Outliers.java";
    String classFileName = "Outliers";

    try {
         // prepareCallObject prepares the Call Object with reqd parameters
         // for calling Algorithm Web Service

        Call call = prepareCallObject(qNameService, qNamePort,
                                      endPoint, nameSpace);

       String[] params = { javaFileName};
        String output = (String) call.invoke(params);
```

```
            // createFile method creates the file in the dirLocation
            createFile(output, javaFileName);
            String data = getDataAsString("T4");

            String executionCommand = javaLocation + " -cp " + dirLocation +
                                      " "+ classFileName + " "+ data + " " +
                                      combinedMean + " " + combinedVariance;
            // executeFile method executes the file and returns the output
            returnValue = executeFile(javaFileName, classFileName,
                                      executionCommand );

        } catch (Exception e) {
            System.out.println(CLASSNAME + methodName + " caught Exception : " +
                               e.toString());
        }

        return returnValue;
    }

    /**
     * Prepares Call Object using the parameters
     * @param qNameService - service name of Algorithm web service
     * @param qNamePort - port of Algorithm web service
     * @param endPoint - end point of Algorithm web service
     * @param nameSpace - name space of Algorithm web service
     * @return  created Call object
     */

    private Call prepareCallObject(String qNameService,
                                   String qNamePort,
                                   String endPoint,
                                   String nameSpace){
        Call call = null;

        String methodName = ".prepareCallObject()";
        System.out.println("Entered " + CLASSNAME + methodName);

        try {
            ServiceFactory factory = ServiceFactory.newInstance();
            Service service = factory.createService(new QName(qNameService));

            QName port = new QName(qNamePort);

            call = service.createCall(port);
            call.setTargetEndpointAddress(endPoint);

            call.setProperty(Call.SOAPACTION_USE_PROPERTY, new Boolean(true));
            call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
            call.setReturnType(QNAME_TYPE_STRING);
            call.setOperationName(new QName(nameSpace, "getAlgorithmFile"));
            call.addParameter("String_1", QNAME_TYPE_STRING, ParameterMode.IN);
        } catch (ServiceException e) {
            System.out.println(CLASSNAME + methodName + " caught
                               ServiceException : " + e.toString());
        }

        return call;
    }
```

```
    /**
     * Creates the source file
     * @param text - text of the file
     * @param fileName - name of the file
     */

    private void createFile(String text, String fileName){
        String methodName=".createFile()";
        System.out.println("Entered " + CLASSNAME + methodName);

        try{
            FileWriter fos = new FileWriter(dirLocation + fileName);
            fos.write(text);
            fos.close();
        } catch (FileNotFoundException e) {
            System.out.println(CLASSNAME + methodName + " caught
                               FileNotFoundException : " + e.toString());
        } catch (IOException e) {
            System.out.println(CLASSNAME + methodName + " caught IOException: "
                               + e.toString());
        }
    }


    /**
     * Compiles, executes and returns the result as String
     * @param javaFileName - name of the java source file
     * @param classFile - name of the class file
     * @param executionCommand - execution command
     * @return - result of executing the class file as String
     */

    private String executeFile(String javaFileName, String classFile, String
executionCommand){
        String temp = "";
        String methodName =".executeFile()";
        String compileCommand = compilerLocation + " " + dirLocation +
                                javaFileName;

        System.out.println("Entered "+ CLASSNAME + methodName);
        System.out.println("Compile Command : " + compileCommand);
        System.out.println("Execution Command : " + executionCommand);
        try {
            Runtime runtime = Runtime.getRuntime();
            Process process = runtime.exec(compileCommand);

            while(! (new File(dirLocation+classFile +".class")).exists()){
                System.out.println(classFile + " Class File doesnt exits");
                Thread.sleep(1000);
            }

            process = runtime.exec(executionCommand);
            BufferedReader bufferedReader = new BufferedReader(new
                    InputStreamReader(process.getInputStream()));

            String out = "";
            StringBuffer sb = new StringBuffer();
            sb.append(bufferedReader.readLine());
            sb.append("\n");
            while ((out = bufferedReader.readLine()) != null) {
                sb.append(out);
                sb.append("\n");
            }
```

```
            temp = sb.toString();
        } catch (IOException e) {
            System.out.println(CLASSNAME + methodName + " caught IOException :
                               " +e.toString());
        } catch (InterruptedException e) {
            System.out.println(CLASSNAME + methodName + " caught
                               InterruptedException : " +e.toString());
        }
       return temp;
}


/**
 * Gets the data from table as comma separated String
 * @param table - name of the table
 * @return comma separated String
 */

private String getDataAsString(String table){
    String methodName=".getDateAsString()";
    System.out.println("Entered " + CLASSNAME + methodName);
    StringBuffer sb = new StringBuffer();

    String query = "SELECT * FROM "+table;
    Connection connection = null;
    Statement statement = null;
    ResultSet rs;

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        connection=DriverManager.getConnection("jdbc:odbc:thesis"," "," ");
        statement = connection.createStatement();

        rs = statement.executeQuery(query);
        while(rs.next()){
            sb.append(rs.getDouble(2));
            sb.append(",");
        }
    } catch (SQLException e) {
        System.out.println("SQLException in " + CLASSNAME + methodName + "
                           : " + e.toString());
    } catch (ClassNotFoundException e) {
        System.out.println("ClassNotFoundException in " + CLASSNAME +
                           methodName + " : " + e.toString());
    } catch (Exception e) {
        System.out.println("Exception in " + CLASSNAME + methodName + " : "
                          + e.toString());
    }finally{
        if(statement != null){
            try {
                statement.close();
            } catch (SQLException e) { /* ignore */ }
        }
        if(connection != null){
            try {
                connection.close();
            } catch (SQLException e) { /* ignore */    }
        }
    }
    return sb.toString();
}
```

```
/**
 * Gets the data as XML String
 * @param table - name of the table
 * @return - String in XML format
 */

public String getData(String table){
    String temp = "";
    String methodName = ".getData()";

    try {
        System.out.println("Entered " + CLASSNAME + methodName);
        temp = getDataFromTableAsXML("T4");
    } catch (Exception e) {
        System.out.println(CLASSNAME + methodName + " caught Exception : "
                           + e.toString());
    }

    return temp;
}


/**
 * Gets data from DB and wraps it in an XML format
 * @param table - name of the table
 * @return String in XML format
 */

private String getDataFromTableAsXML(String table){
    String methodName=".getDateFromTableAsXML()";
    System.out.println("Entered " + CLASSNAME + methodName);
    String output = null;
    String query = "SELECT * FROM "+table;

    Connection connection = null;
    Statement statement = null;
    ResultSet rs;

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        connection=DriverManager.getConnection("jdbc:odbc:thesis", "","");
        statement = connection.createStatement();

        rs = statement.executeQuery(query);
        Element root = new Element("data");
        while (rs.next()) {
            Element row = new Element("row");
            Element value = new Element("value");
            value.setText(String.valueOf(rs.getDouble(2)));
            row.addContent(value);
            root.addContent(row);
        }
        XMLOutputter serializer = new XMLOutputter();
        serializer.setIndent("  ");
        serializer.setNewlines(true);
        output = serializer.outputString(root);
    } catch (SQLException e) {
        System.out.println(CLASSNAME + methodName + " caught SQLException :
                           " + e.toString());
    } catch (ClassNotFoundException e) {
        System.out.println(CLASSNAME + methodName + " caught
                           ClassNotFoundException : " + e.toString());
    } catch (Exception e) {
```

```
            System.out.println(CLASSNAME + methodName + " caught Exception : "
                               + e.toString());
        }finally{
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException e) { /* ignore */ }
            }
            if(connection != null){
                try {
                    connection.close();
                } catch (SQLException e) { /* ignore */ }
            }
        }
        return output;
    }
}
```

Registries.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: Registries.java
* Purpose: Used to retrieve all the registry server names and URLs from
           Database
*/

package webclient;

import java.sql.*;
import java.util.LinkedList;
import java.util.List;

public class Registries {
    private static String CLASSNAME="webclient.Registries";

    public Registries(){

    }

    /**
     * gets all the Registries from the DB
     * @return List of Registry Objects
     */

    public List getRegistriesFromDB(){
        List temp = new LinkedList();
        String methodName = ".getRegistriesFromDB()";
        String query = "SELECT * FROM RegistryTable";

        Connection connection = null;
        Statement statement = null;
        ResultSet rs;

        Registry registry = null;
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            connection=DriverManager.getConnection("jdbc:odbc:thesis"," "," ");
            statement = connection.createStatement();

            rs = statement.executeQuery(query);
            while(rs.next()){
                registry = new Registry();
                registry.setName(rs.getString("name"));
                registry.setUrl(rs.getString("url"));
                temp.add(registry);
            }
        } catch (SQLException e) {
            System.out.println("SQLException in " + CLASSNAME + methodName + "
                               : " + e.toString());
        } catch (ClassNotFoundException e) {
            System.out.println("ClassNotFoundException in " + CLASSNAME +
                               methodName + " : " + e.toString());
        } catch (Exception e) {
            System.out.println("Exception in " + CLASSNAME + methodName + " : "
+ e.toString());
        }finally{
            if(statement != null){
```

```
                try {
                    statement.close();
                } catch (SQLException e) { /* ignore */ }
            }
            if(connection != null){
                try {
                    connection.close();
                } catch (SQLException e) {/* ignore */ }
            }
        }
        return temp;
    }

}
```

Registry.java

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: Registry.java
* Purpose: Registry Bean used to store information about Registry Server
           Used by WebClient and Client to display Registry Server Information
*/

package webclient;

public class Registry {
    private String url;
    private String name;

    public String getUrl() {
        return url;
    }

    public void setUrl(String URL) {
        this.url = URL;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String toString() {
        return "Registry{" +
                "URL='" + url + "'" +
                ", name='" + name + "'" +
                "}";
    }
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: JAXQuery.java
* Purpose: Class used to query Registry Server and return the Algorithm Web
*          Services and Data Web Service
*/

package client;

import javax.xml.registry.*;
import javax.xml.registry.infomodel.*;
import java.util.*;

/**
 * The JAXRQuery class consists of a main method, a
 * makeConnection method, an executeQuery method, and some
 * private helper methods. It searches a registry for
 * information about organizations whose names contain a
 * user-supplied string.
 */
public class JAXRQuery {
    Connection connection = null;

    /**
     * public constructor
     */

    public JAXRQuery() {}


    /**
     * Queries the registry specified by the queryURL for organization names
     * that match String arg
     * @param arg - String to search for
     * @param queryURL - Registry Server URL
     * @return List of JAXRQueryResult Objects
     */

    public List getDataMiningServices(String arg, String queryURL) {

        String queryString = new String(arg);
        System.out.println("Query string is ' " + queryString + " ' ");
        System.out.println("Query URL : " + queryURL);
        System.out.println();

        JAXRQuery jq = new JAXRQuery();
        jq.makeConnection(queryURL);

        return jq.executeQuery(queryString);
    }


    /**
     * Establishes a connection to a registry.
     * @param queryUrl     the URL of the query registry
     */

    public void makeConnection(String queryUrl) {
```

```
        /*
         * Specify proxy information in case you
         *  are going beyond your firewall.
         */
         String httpProxyHost = "";
         String httpProxyPort = "8080";
        /*
         * Define connection configuration properties.
         * For simple queries, you need the query URL.
         */
        Properties props = new Properties();
        props.setProperty("javax.xml.registry.queryManagerURL",
            queryUrl);
        props.setProperty("com.sun.xml.registry.http.proxyHost",
            httpProxyHost);
        props.setProperty("com.sun.xml.registry.http.proxyPort",
            httpProxyPort);


        try {
            // Create the connection, passing it the
            // configuration properties
            ConnectionFactory factory = ConnectionFactory.newInstance();
            factory.setProperties(props);
            connection = factory.createConnection();
            System.out.println("Created connection to registry");
        } catch (Exception e) {
            e.printStackTrace();
            if (connection != null) {
                try {
                    connection.close();
                } catch (JAXRException je) {}
            }
        }
    }


    /**
     * Searches for organizations containing a string and
     * displays data about them.
     * @param qString      the string argument
     */

    public List executeQuery(String qString) {
        RegistryService rs = null;
        BusinessQueryManager bqm = null;
        Collection orgs = null;
        JAXRQueryResult queryResult;
        List JAXQueryResultList = new LinkedList();
        try {
            // Get registry service and query manager
            rs = connection.getRegistryService();
            bqm = rs.getBusinessQueryManager();
            System.out.println("Got registry service and " +
                "query manager");

            // Define find qualifiers and name patterns
            Collection findQualifiers = new ArrayList();
            findQualifiers.add(FindQualifier.SORT_BY_NAME_ASC);
            Collection namePatterns = new ArrayList();
            namePatterns.add("%" + qString + "%");
```

```
        // Find using the name
        BulkResponse response = bqm.findOrganizations(findQualifiers,
                namePatterns, null, null, null, null);
        orgs  = response.getCollection();
        // Display information about the organizations found
        Iterator orgIter = orgs.iterator();
        if (!(orgIter.hasNext())) {
            System.out.println("No organizations found");
        } else while (orgIter.hasNext()) {

            Organization org = (Organization) orgIter.next();

            // Display primary contact information
            User pc = org.getPrimaryContact();
            if (pc != null) {
                PersonName pcName = pc.getPersonName();
                Collection phNums = pc.getTelephoneNumbers(null);
                Iterator phIter = phNums.iterator();
                while (phIter.hasNext()) {
                    TelephoneNumber num = (TelephoneNumber) phIter.next();
                }
                Collection eAddrs = pc.getEmailAddresses();
                Iterator eaIter = eAddrs.iterator();
                while (eaIter.hasNext()) {
                    EmailAddress eAd =  (EmailAddress) eaIter.next();
                }
            }
            // Display service and binding information
            Collection services = org.getServices();
            Iterator svcIter = services.iterator();
            while (svcIter.hasNext()) {
                Service svc = (Service) svcIter.next();
                if((getServiceName(svc )!= null) ||
                   (getServiceDescription(svc) != null)){

                    queryResult = new JAXRQueryResult();
                    queryResult.setServiceName(getServiceName(svc));

                    queryResult.setServiceDescription(
                                    getServiceDescription(svc));
                    Collection serviceBindings = svc.getServiceBindings();
                    Iterator sbIter = serviceBindings.iterator();
                    while (sbIter.hasNext()) {
                        ServiceBinding sb = (ServiceBinding) sbIter.next();
                        queryResult.setServiceBindingDesc(
                                        getServiceDescription(sb));
                        ueryResult.setServiceBindingURI(sb.getAccessURI());
                    }
                    // Print spacer between organizations
                    JAXQueryResultList.add(queryResult);
                }
            }

        }
    } catch (Exception e) {
        System.out.println("Exception executeQuery() : " + e.toString());
    } finally  {
        // At end, close connection to registry
        if (connection != null) {
            try {
                connection.close();
            } catch (JAXRException je) {}
        }
```

```
        }
        return JAXQueryResultList;
    }


    /**
     * Returns the name value for a registry object.
     * @param ro    a RegistryObject
     * @return              the String value
     */

    private String getServiceName(RegistryObject ro)
        throws JAXRException {

        try {
            return ro.getName().getValue();
        } catch (NullPointerException npe) {
            return "No Name";
        }
    }


    /**
     * Returns the description value for a registry object.
     *
     * @param ro    a RegistryObject
     * @return              the String value
     */

    private String getServiceDescription(RegistryObject ro)
        throws JAXRException {

        try {
            return ro.getDescription().getValue();
        } catch (NullPointerException npe) {
            return "No Description";
        }
    }

}
```

JAXRQueryResult.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: JAXRQueryResult.java
* Purpose: JAXRQueryResult Class saves the information about the Web Service
*          for display in WebClient and command line inteface
*/

package client;

public class JAXRQueryResult {
    private String serviceName;
    private String serviceDescription;
    private String serviceBindingDesc;
    private String serviceBindingURI;

    public JAXRQueryResult() {
    }

    public String getServiceName() {
        return serviceName;
    }

    public void setServiceName(String serviceName) {
        this.serviceName = serviceName;
    }

    public String getServiceDescription() {
        return serviceDescription;
    }

    public void setServiceDescription(String serviceDescription) {
        this.serviceDescription = serviceDescription;
    }

    public String getServiceBindingDesc() {
        return serviceBindingDesc;
    }

    public void setServiceBindingDesc(String serviceBindingDesc) {
        this.serviceBindingDesc = serviceBindingDesc;
    }

    public String getServiceBindingURI() {
        return serviceBindingURI;
    }

    public void setServiceBindingURI(String serviceBindingURI) {
        this.serviceBindingURI = serviceBindingURI;
    }

}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: WSDLParser.java
* Purpose: Used to parser the WSDL file for a web service and return it as
*          WSDLObject Class
*/

package client;

import org.jdom.Attribute;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.List;

public class WSDLParser {
    private WSDLObject wsdlObject;

    public WSDLParser() {
    }

    public WSDLObject getWsdlObject() {
        return wsdlObject;
    }

    public void setWsdlObject(WSDLObject wsdlObject) {
        this.wsdlObject = wsdlObject;
    }

    /**
     * Parses the WSDL file and creates a WSDLObject
     * @param wsdlFile - WSDL File
     */

    public void parseWSDLFile(String wsdlFile){
        String methodName = "parseWSDLFile()";
        SAXBuilder  builder;
        Document    doc     = null;
        Element     root    = null;
        URL wsdlURL = null;
        WSDLObject wsdlObject;
        try {
            wsdlURL = new URL(wsdlFile);

            builder = new SAXBuilder();
            doc =  builder.build(wsdlURL);
            root = doc.getRootElement();

            wsdlObject = new WSDLObject();

            Attribute nameSpace = root.getAttribute("targetNamespace");
            wsdlObject.setNameSpace(nameSpace.getValue());

            List childerList = root.getChildren();
```

```
            for (int i = 0; i < childerList.size(); i++) {
                Element element = (Element) childerList.get(i);

                if(element.getName().equalsIgnoreCase("portType")){
                    Attribute attribute = element.getAttribute("name");
                    wsdlObject.setPort(attribute.getValue());

                    List opertionList = element.getChildren();
                    for (int j = 0; j < opertionList.size(); j++) {

                        Operation operationObject = new Operation();

                        Element operation = (Element) opertionList.get(j);

                        String operationName =
                                operation.getAttributeValue("name");
                        operationObject.setName(operationName);

                        String parameters =
                               operation.getAttributeValue("parameterOrder");
                        if (parameters != null) {
                            String parameterOrder[] = parameters.split("\\s");
                            operationObject.setParameterOrder(parameterOrder);
                        }


                        wsdlObject.setOperation(operationObject);

                    }
                }else if(element.getName().equalsIgnoreCase("service")){
                    Attribute attribute = element.getAttribute("name");
                    wsdlObject.setService(attribute.getValue());
                    List serviceChildren = element.getChildren();

                    if(serviceChildren.size()>0) {
                        Element port = (Element) serviceChildren.get(0);
                        List soap = port.getChildren();
                        if(soap.size() > 0){
                            Element soapAddress = (Element) soap.get(0);
                            Attribute location =
                                    soapAddress.getAttribute("location");
                            wsdlObject.setEndPoint(location.getValue());
                        }
                    }
                }
            }
            this.wsdlObject = wsdlObject;

        } catch (MalformedURLException e) {
            System.out.println("MalformedURLException in " + methodName + " :
                               "+e.toString());
        } catch (JDOMException e) {
            System.out.println("JDOMException in " + methodName + " : " +
                               e.toString());
        }
    }

}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: WSDLObject.java
* Purpose: Used to save information regarding a Web Service
*/

package client;

public class WSDLObject {
    private String service;
    private String port;
    private String nameSpace;
    private Operation operation;
    private String endPoint;

    public WSDLObject() {
    }

    public String getEndPoint() {
        return endPoint;
    }

    public void setEndPoint(String endPoint) {
        this.endPoint = endPoint;
    }

    public Operation getOperation() {
        return operation;
    }

    public void setOperation(Operation operation) {
        this.operation = operation;
    }

    public String getPort() {
        return port;
    }

    public void setPort(String port) {
        this.port = port;
    }

    public String getNameSpace() {
        return nameSpace;
    }

    public void setNameSpace(String nameSpace) {
        this.nameSpace = nameSpace;
    }

    public String getService() {
        return service;
    }

    public void setService(String service) {
        this.service = service;
    }
```

```
    public String toString() {
        return "WSDLObject{" +
                "service='" + service + "'" +
                ", port='" + port + "'" +
                ", nameSpace='" + nameSpace + "'" +
                ", operation=" + operation +
                ", endPoint='" + endPoint + "'" +
                "}";
    }
}
```

ResultMeanVariance.java

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: ResultMeanVariance.java
* Purpose: Mean Varaince calculated using Web Services is saved in this object
*/

package client;

public class ResultMeanVariance {
    private int noOfElements;
    private double mean;
    private double variance;

    public int getNoOfElements() {
        return noOfElements;
    }

    public void setNoOfElements(int noOfElements) {
        this.noOfElements = noOfElements;
    }

    public double getMean() {
        return mean;
    }

    public void setMean(double mean) {
        this.mean = mean;
    }

    public double getVariance() {
        return variance;
    }

    public void setVariance(double variance) {
        this.variance = variance;
    }

    public String toString() {
        return "No Of Elements=" + noOfElements +"\n"+
               "mean=" + mean + "\n"+
               "variance=" + variance ;
    }
}
```

Outlier.java

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: Outlier.java
* Purpose: Outliers calculated from Web Services is saved in this object
*/

package client;

public class Outlier {
    private String dataSet;
    private String range;
    private String outliers;

    public Outlier() {
    }

    public String getDataSet() {
        return dataSet;
    }

    public void setDataSet(String dataSet) {
        this.dataSet = dataSet;
    }

    public String getRange() {
        return range;
    }

    public void setRange(String range) {
        this.range = range;
    }

    public String getOutliers() {
        return outliers;
    }

    public void setOutliers(String outliers) {
        this.outliers = outliers;
    }

    public String toString() {
        return dataSet + "\n" +
               range + "\n" +
               outliers + "\n";
    }
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: Client.java
* Purpose: Main Class in Command Line Interface. Displays the Registries, Runs
*          the web service and displays the results, uses other classes
*/

package client;

import javax.xml.namespace.QName;
import javax.xml.rpc.*;
import java.io.*;
import java.rmi.RemoteException;
import java.util.*;

public class Client {

    private static double combinedMean = 0;
    private static double combinedVariance = 0;
    private static String CLASSNAME="Client";

    private static String BODY_NAMESPACE_VALUE = "urn:Foo";
    private static String ENCODING_STYLE_PROPERTY =
                          "javax.xml.rpc.encodingstyle.namespace.uri";
    private static String NS_XSD = "http://www.w3.org/2001/XMLSchema";
    private static String URI_ENCODING =
                           "http://schemas.xmlsoap.org/soap/encoding/";

    public static void main(String[] args) {
        String queryURL = "";
        String algorithmString ="algorithmwebservice";
        String dataString="dataservices";
        Registries registries = new Registries();
        List registryList = registries.getRegistriesFromDB();
        queryURL = displayRegistries(registryList);

        String meanVarianceWSDLFile = "";
        String outlierWSDLFile = "";

        JAXRQuery jaxrQuery = new JAXRQuery();
        WSDLParser wsdlParser = new WSDLParser();

        // Query the Registry for algorithm web services
        List algorithmOrgs = jaxrQuery.getDataMiningServices(algorithmString,
                                                             queryURL);

        displayHeading("Algorithm Web Services");

        for (int i = 0; i < algorithmOrgs.size(); i++) {
            JAXRQueryResult jaxrQueryResult =
                            (JAXRQueryResult) algorithmOrgs.get(i);
            if(jaxrQueryResult.getServiceDescription().indexOf("Mean") != -1){
             meanVarianceWSDLFile = jaxrQueryResult.getServiceBindingURI()
                                    +"?WSDL";
            }else{
                outlierWSDLFile = jaxrQueryResult.getServiceBindingURI()
                                  +"?WSDL";
            }
```

```
            displayJAXRQueryResult(i, jaxrQueryResult);
        }
        System.out.println();

        // Query the Registry for data web services
        List dataOrgs = jaxrQuery.getDataMiningServices(dataString, queryURL);
        List dataMap = new LinkedList();

        displayHeading("Data Web Services");

        for (int i = 0; i < dataOrgs.size(); i++) {
            JAXRQueryResult jaxrQueryResult =
                            (JAXRQueryResult) dataOrgs.get(i);
            String dataWSDLFile =
                           jaxrQueryResult.getServiceBindingURI()+"?WSDL";
            displayJAXRQueryResult(i, jaxrQueryResult);

            // parse the WSDL files and save the wsdlobject in a List
            wsdlParser.parseWSDLFile(dataWSDLFile);
            dataMap.add(wsdlParser.getWsdlObject());
            System.out.println();
        }

        displayDottedLines();

        WSDLObject mvAlgoWsdlObject = null;
        WSDLObject outlierAlgoWsdlObject = null;
        List resultMeanVarianceList = new LinkedList();

        if(meanVarianceWSDLFile != null
           && meanVarianceWSDLFile.trim().length() > 0){
            wsdlParser.parseWSDLFile(meanVarianceWSDLFile);
            mvAlgoWsdlObject = wsdlParser.getWsdlObject();
        }

        if(outlierWSDLFile != null && outlierWSDLFile.trim().length() > 0){
            wsdlParser.parseWSDLFile(outlierWSDLFile);
            outlierAlgoWsdlObject = wsdlParser.getWsdlObject();
        }

        int choice = displayExecutionOptions();

        if(choice == 1){
            // Algorithm to Data
            algorithmToData(dataMap, mvAlgoWsdlObject,
                            resultMeanVarianceList, outlierAlgoWsdlObject);
        }else if(choice == 2){
            //Data to Algorithm
            dataToAlgorithm(dataMap, mvAlgoWsdlObject,
                            resultMeanVarianceList, outlierAlgoWsdlObject);
        }else{
            //Data to Algorithm
            dataToAlgorithm(dataMap, mvAlgoWsdlObject,
                            resultMeanVarianceList, outlierAlgoWsdlObject);
            // Algorithm to Data
            algorithmToData(dataMap, mvAlgoWsdlObject,
                            resultMeanVarianceList, outlierAlgoWsdlObject);

        }

    }
```

```
private static void displayHeading(String heading) {
    System.out.println();
    System.out.println("---------- "+ heading +" ---------- ");
    System.out.println();
}

private static int displayExecutionOptions() {
    int choice = 0;

    displayHeading("Execution Options");

    System.out.println("1. Algorithm To Data");
    System.out.println("2. Data To Algorithm ");
    System.out.println("3. Both");
    System.out.println();
    try {
        do{
            System.out.print("Please select an option : ");
            choice = getIntegerInput();
        }while(choice <1 || choice > 3);
    } catch (IOException e) {
        System.out.println("IOException : " + e.toString());
    }
    return choice;
}


/**
 * Runs Algorithm web service. It passes the data web service information
 * to algorithm web service
 * @param dataMap - List of data web service objects
 * @param mvAlgoWsdlObject-Mean Variance Algorithm
 *                          Webservice information object
 * @param resultMeanVarianceList - Mean Variance Object List
 * @param outlierAlgoWsdlObject - Outlier Algorithm
 *                                Web Service information object
 */

private static void dataToAlgorithm(List dataMap,
                                    WSDLObject mvAlgoWsdlObject,
                                    List resultMeanVarianceList,
                                    WSDLObject outlierAlgoWsdlObject) {
    Date start;
    Date end;
    System.out.println();
    System.out.println("-------------- Data to Algorithm -------------");
    System.out.println();
    start = new Date();
    for (int i = 0; i < dataMap.size(); i++) {
        WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
        System.out.println("Running " + dataWSDLObject.getService() +
                           " on MeanVariance Web Service");
        System.out.println();
        ResultMeanVariance resultMeanVariance =
                 getDataRunMeanVariance(mvAlgoWsdlObject, dataWSDLObject);
        resultMeanVarianceList.add(resultMeanVariance);
        System.out.println(resultMeanVariance);
        System.out.println();
    }

    combinedMean = combinedVariance = 0;
    if(combinedMean == 0 || combinedVariance == 0){
       System.out.println("Calculating CombinedMean and CombinedVariance");
```

```
            getCombinedMeanVariance(resultMeanVarianceList);
        }

        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running " + dataWSDLObject.getService() +
                               " on Outlier Web Service");
            System.out.println();
            System.out.println(getDataRunOutliers(outlierAlgoWsdlObject,
                                                  dataWSDLObject));
            System.out.println();
        }
        System.out.println();
        end = new Date();
        System.out.println("start : "+ start);
        System.out.println("end : " +end);
    }


    /**
     * Runs Data web service. It passes the algorithm web service information
     * to data web service
     * @param dataMap - List of data web service objects
     * @param mvAlgoWsdlObject - Mean Variance Algorithm Web service
     *                           information object
     * @param resultMeanVarianceList - Mean Variance Object List
     * @param outlierAlgoWsdlObject - Outlier Algorithm Web Service
     *                                information object
     */

    private static void algorithmToData(List dataMap,
                                        WSDLObject mvAlgoWsdlObject,
                                        List resultMeanVarianceList,
                                        WSDLObject outlierAlgoWsdlObject) {
        System.out.println();
        System.out.println("------------ Algorithm to Data ----------------");
        System.out.println();
        Date start = new Date();
        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running MeanVariance Web Service on Data "
                               + (i+1));
            ResultMeanVariance resultMeanVariance =
                        getMeanVarianceForData(dataWSDLObject, mvAlgoWsdlObject);
            resultMeanVarianceList.add(resultMeanVariance);
            System.out.println();
            System.out.println(resultMeanVariance);
            System.out.println();
        }

        if(combinedMean == 0 || combinedVariance == 0){
            System.out.println("Calculating CombinedMean and
                                CombinedVariance");
            getCombinedMeanVariance(resultMeanVarianceList);
        }
        System.out.println();
        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running Outlier Web Service on Data " + (i+1));
            System.out.println(getOutliersForData(dataWSDLObject,
                        outlierAlgoWsdlObject, combinedMean, combinedVariance));
        }
        Date end = new Date();
```

```
        System.out.println("start : "+ start);
        System.out.println("end : " +end);
        System.out.println();
    }


    /**
     * Displays the avialable registry servers
     * @param registryList
     * @return
     */

    private static String displayRegistries(List registryList) {
        System.out.println();
        System.out.println("---------- Registries ---------- ");
        System.out.println();

        String queryURL = "";

        for (int i = 0; i < registryList.size(); i++) {
            Registry registry = (Registry) registryList.get(i);
            System.out.println((i+1) + ". " + registry.getName());
        }
        System.out.println();

        try {
            int choice = 0;
            do{
                System.out.print("Select a Registry : ");
                choice = getIntegerInput();
            }while(choice < 1 || choice > (registryList.size()+1) );

            for (int i = 0; i < registryList.size(); i++) {
                if(choice == (i+1)){
                    Registry registry = (Registry) registryList.get(i);
                    queryURL = registry.getUrl();
                }
            }

        } catch (IOException e) {
            System.out.println("IOException in displayRegistries() : " +
                               e.toString());
        }
        System.out.println();
        return queryURL;
    }


    /**
     * Displays dotted lines for clarity and neatness
     */

    private static void displayDottedLines() {
        System.out.println();
        System.out.println("----------------------------------------- ");
        System.out.println();
    }


    /**
     * Prints the Result from Registry onto the Screen
     * @param i
     * @param jaxrQueryResult - object returned from JAXQuery
```

```
 */

private static void displayJAXRQueryResult(int i,
                                            JAXRQueryResult jaxrQueryResult) {
    System.out.println((i+1)+"."+jaxrQueryResult.getServiceDescription() );
    System.out.println("\tURI :" + jaxrQueryResult.getServiceBindingURI());
    System.out.println("\tWSDL File : " +
                       jaxrQueryResult.getServiceBindingURI()+"?WSDL");
    System.out.println();
}


/**
 * Gets the input from console and converts to int
 * @return integer
 * @throws IOException
 */

private static int getIntegerInput() throws IOException {
    BufferedReader bufferedReader = null;
    int temp = 0;

    bufferedReader = new BufferedReader(new InputStreamReader(System.in));
    String input = bufferedReader.readLine();
    temp = Integer.parseInt(input.trim());

    return temp;
}


/**
 * Runs the webservice by calling callMeanVarianceForData,
 * prepares the result as an ResultMeanVariance object
 * @param dataWsdlObject
 * @param algorithmWsdlObject
 * @return ResultMeanVariance object
 */

public static ResultMeanVariance getMeanVarianceForData(
                                            WSDLObject dataWsdlObject,
                                            WSDLObject algorithmWsdlObject){
    ResultMeanVariance resultMeanVariance = null;

    try {
        String output1 = callMeanVarianceForData(dataWsdlObject,
                                                 algorithmWsdlObject);
        String str1[] = new String[3];
        StringTokenizer stoken1 = new StringTokenizer(output1, ",");
        int i1=0;
        while(stoken1.hasMoreElements())
            str1[i1++] = stoken1.nextToken();
            resultMeanVariance = new ResultMeanVariance();
            Double I1 = new Double(str1[0]);
            resultMeanVariance.setNoOfElements(I1.intValue());
            Double dm1 = new Double(str1[1]);
            resultMeanVariance.setMean(dm1.doubleValue());
            Double dv1 = new Double(str1[2]);
            resultMeanVariance.setVariance(dv1.doubleValue());
    }catch(Exception e){
        e.printStackTrace();
    }
    return resultMeanVariance;
}
```

```
/**
 * This actually runs the webservice by passing the Algorithm parameters to
 * Data web service. Data web service calls the algorithm webservice using
 * the algorithm parameters. result is returned as String
 * @param dataWsdlObject - Data Web Service parameters
 * @param algorithmWsdlObject - Algorithm Web Service parameters
 * @return String returned by Data Web Service running Algorithm Web
 *         Service on its data
 */

public static String callMeanVarianceForData(WSDLObject dataWsdlObject,
                                              WSDLObject algorithmWsdlObject){

    QName QNameString = new QName(NS_XSD, "string");

    String temp ="";
    String dataEndPoint = dataWsdlObject.getEndPoint();
    String dataQNameService = dataWsdlObject.getService();
    String dataQNamePort = dataWsdlObject.getPort();
    String dataNameSpace = dataWsdlObject.getNameSpace();

    String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
    String algorithmQNameService = algorithmWsdlObject.getService();
    String algorithmQNamePort = algorithmWsdlObject.getPort();
    String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

    ServiceFactory factory = null;
    try {
        factory = ServiceFactory.newInstance();
    } catch (ServiceException e) {
        System.out.println("ServiceException in callMeanVarianceForData for
                        Service " +dataQNameService +" : " + e.toString());
    }

    String ns = BODY_NAMESPACE_VALUE;
    if(dataNameSpace != null){
        ns = dataNameSpace;
    }

    try {
        Service service = factory.createService(new
                                    QName(dataQNameService));
        QName port = new QName(dataQNamePort);

        Call call = service.createCall(port);
        call.setTargetEndpointAddress(dataEndPoint);

        call.setProperty(Call.SOAPACTION_USE_PROPERTY,
            new Boolean(true));
        call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
        call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

        call.setReturnType(QNameString);
        call.setOperationName(new QName(ns, "getMeanVariance"));
        call.addParameter("String_1", QNameString, ParameterMode.IN);
        call.addParameter("String_2", QNameString, ParameterMode.IN);
        call.addParameter("String_3", QNameString, ParameterMode.IN);
        call.addParameter("String_4", QNameString, ParameterMode.IN);
        String[] param = {algorithmEndPoint, algorithmQNameService,
                          algorithmQNamePort, algorithmNameSpace};
```

```
            temp = (String) call.invoke(param);
        } catch (ServiceException e) {
            System.out.println("ServiceException in callMeanVarianceForData1
                        for Service " + dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println("RemoteException in callMeanVarianceForData1 " +
                    dataQNameService +" : " + e.toString());
        } catch (IOException e) {
            System.out.println("IOException in callMeanVarianceForData1 " +
                    dataQNameService +" : " + e.toString());
        }
        return temp ;
    }


    /**
     * Gets the comined mean and combined variance from ResultMeanVarianceList
     * @param resultMeanVarianceList
     */

    public static void getCombinedMeanVariance(List resultMeanVarianceList){
           int count1,count2;
           double mean1,variance1,mean2,variance2;

            if(resultMeanVarianceList.size()>1){
                ResultMeanVariance resultMeanVariance =
                      (ResultMeanVariance) resultMeanVarianceList.get(0);
                count1 = resultMeanVariance.getNoOfElements();
                mean1 = resultMeanVariance.getMean();
                variance1 = resultMeanVariance.getVariance();
                resultMeanVariance =
                     (ResultMeanVariance) resultMeanVarianceList.get(1);
                count2 = resultMeanVariance.getNoOfElements();
                mean2 = resultMeanVariance.getMean();
                variance2 = resultMeanVariance.getVariance();

                combinedMeanVariance(mean1, variance1, count1,
                        mean2, variance2, count2);

            }
        }


    /**
     * helper method that calculates the combined mean and combined variance
     * @param dmean1
     * @param dvar1
     * @param ct1
     * @param dmean2
     * @param dvar2
     * @param ct2
     */

    private static void combinedMeanVariance(double dmean1,double dvar1,
                                             int ct1, double dmean2,
                                             double dvar2, int ct2) {
        double cmean = 0, cvar = 0, cvarsqr = 0;
        cmean = (((dmean1 * ct1) + (dmean2 * ct2))/(ct1 + ct2));
        cvarsqr=(((dvar1*dvar1)/(ct1))+((dvar2 * dvar2)/ (ct2))) *(ct1+ct2);
        cvar=Math.sqrt(cvarsqr);
        combinedMean = cmean;
        combinedVariance = cvar;
    }
```

```
/**
 * Gets the Outlier Object by calling callOutliersForData method
 * @param dataWsdlObject
 * @param algorithmWsdlObject
 * @param cmean
 * @param cvar
 * @return
 */

public static Outlier getOutliersForData(WSDLObject dataWsdlObject,
                                         WSDLObject algorithmWsdlObject,
                                         double cmean,
                                         double cvar){
    String output = null;
    Outlier outlier = new Outlier();
    try {
        System.out.println();
        output = callOutliersForData(dataWsdlObject, algorithmWsdlObject,
                                     cmean, cvar);
        String str1[] = new String[3];
        StringTokenizer tokens = new StringTokenizer(output, "|");
        int i1=0;
        while(tokens.hasMoreElements())
            str1[i1++] = tokens.nextToken();
            outlier.setDataSet(str1[0]);
            outlier.setRange(str1[1]);
            outlier.setOutliers(str1[2]);
    } catch (Exception e) {
        System.out.println("Exception in
                webclient.Client.getOutliersForData() : " + e.toString());
    }
    return outlier;
}


/**
 * helper method - this method actually calls the data web service
 * passing to it the algorithm web service parameters
 * @param dataWsdlObject - Data Web Service parameters
 * @param algorithmWsdlObject - Algorithm Web Service parameters
 * @param combinedMean
 * @param combinedVariance
 * @return the result from data web service
 */

public static String callOutliersForData(WSDLObject dataWsdlObject,
                                         WSDLObject algorithmWsdlObject,
                                         double combinedMean,
                                         double combinedVariance){
    QName QNameString = new QName(NS_XSD, "string");

    String temp="";
    String dataEndPoint = dataWsdlObject.getEndPoint();
    String dataQNameService = dataWsdlObject.getService();
    String dataQNamePort = dataWsdlObject.getPort();
    String dataNameSpace = dataWsdlObject.getNameSpace();

    String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
    String algorithmQNameService = algorithmWsdlObject.getService();
    String algorithmQNamePort = algorithmWsdlObject.getPort();
    String algorithmNameSpace = algorithmWsdlObject.getNameSpace();
```

```
        System.out.println("Running Outlier Web Service on ");
        System.out.println("Endpoint : " +  dataEndPoint);

        ServiceFactory factory = null;
        try {
            factory = ServiceFactory.newInstance();
        } catch (ServiceException e) {
            System.out.println("ServiceException in callMeanVarianceForData for
                        Service " + dataQNameService +" : " + e.toString());
        }
        String ns = BODY_NAMESPACE_VALUE;
        if(dataNameSpace != null){
            ns = dataNameSpace;
        }

        try {
            Service service = factory.createService(new
                                                QName(dataQNameService));
            QName port = new QName(dataQNamePort);

            Call callOutlier = service.createCall(port);
            callOutlier.setTargetEndpointAddress(dataEndPoint);

            callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
                new Boolean(true));
            callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            callOutlier.setReturnType(QNameString);
            callOutlier.setOperationName(new QName(ns, "getOutliers"));
            callOutlier.addParameter("String_1",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_2",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_3",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_4",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_5",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_6",QNameString, ParameterMode.IN);
            String[] params = { algorithmEndPoint, algorithmQNameService,
                                algorithmQNamePort, algorithmNameSpace,
               String.valueOf(combinedMean), String.valueOf(combinedVariance)};

            temp = (String)callOutlier.invoke(params);
        } catch (ServiceException e) {
            System.out.println("ServiceException in callOutliersForData " +
                    dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println("RemoteException in callOutliersForData " +
                    dataQNameService +" : " + e.toString());
        } catch(Exception e){
            e.printStackTrace();
        }
        return temp;
    }


    /**
     * Runs the algorithm web service passing to it the data web
     * service parameters
     * @param algorithmWsdlObject - Algorithm Web Service parameters
     * @param dataWsdlObject - Data web service parameters
     * @return ResultMeanVariance Object
     */

    public static ResultMeanVariance getDataRunMeanVariance(
```

```
        WSDLObject algorithmWsdlObject, WSDLObject dataWsdlObject){

QName QNameString = new QName(NS_XSD, "string");

String temp="";
ResultMeanVariance resultMeanVariance = null;
String methodName = ".getDataRunMeanVariance()";

String dataEndPoint = dataWsdlObject.getEndPoint();
String dataQNameService = dataWsdlObject.getService();
String dataQNamePort = dataWsdlObject.getPort();
String dataNameSpace = dataWsdlObject.getNameSpace();

String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
String algorithmQNameService = algorithmWsdlObject.getService();
String algorithmQNamePort = algorithmWsdlObject.getPort();
String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

ServiceFactory factory = null;
try {
    factory = ServiceFactory.newInstance();
} catch (ServiceException e) {
    System.out.println(CLASSNAME + methodName + "caught
                        ServiceException for Service " +
                        dataQNameService +" : " + .toString());
}

String ns = BODY_NAMESPACE_VALUE;
if(algorithmNameSpace != null){
    ns = algorithmNameSpace;
}

try {
    Service service = factory.createService(
                          new QName(algorithmQNameService));
    QName port = new QName(algorithmQNamePort);

    Call callOutlier = service.createCall(port);
    callOutlier.setTargetEndpointAddress(algorithmEndPoint);

    callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
        new Boolean(true));
    callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
    callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

    callOutlier.setReturnType(QNameString);
    callOutlier.setOperationName(
                  new QName(ns, "getDataRunMeanVariance") );
    callOutlier.addParameter("String_1",QNameString, ParameterMode.IN);
    callOutlier.addParameter("String_2",QNameString, ParameterMode.IN);
    callOutlier.addParameter("String_3",QNameString, ParameterMode.IN);
    callOutlier.addParameter("String_4",QNameString, ParameterMode.IN);

    String[] params = {  dataQNameService, dataQNamePort,
                         dataEndPoint, dataNameSpace};

    temp = (String)callOutlier.invoke(params);

    String str1[] = new String[3];
    StringTokenizer stoken1 = new StringTokenizer(temp, ",");
    int i1=0;
    while(stoken1.hasMoreElements())
        str1[i1++] = stoken1.nextToken();
```

```
            resultMeanVariance = new ResultMeanVariance();
            Double I1 = new Double(str1[0]);
            resultMeanVariance.setNoOfElements(I1.intValue());
            Double dm1 = new Double(str1[1]);
            resultMeanVariance.setMean(dm1.doubleValue());
            Double dv1 = new Double(str1[2]);
            resultMeanVariance.setVariance(dv1.doubleValue());
        } catch (ServiceException e) {
            System.out.println( CLASSNAME + methodName + "caught
                  ServiceException " + dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println(CLASSNAME + methodName + "caught
                  ServiceException "+ dataQNameService +" : " + e.toString());
        } catch(Exception e){
            e.printStackTrace();
        }
        return resultMeanVariance;
    }


    /**
     * Runs the Outlier web service passing to it the
     * data web service parameters
     * @param algorithmWsdlObject - Algorithm Web Service parameters
     * @param dataWsdlObject - Data web service parameters
     * @return Outlier Object
     */

    public static Outlier getDataRunOutliers(WSDLObject algorithmWsdlObject,
                                             WSDLObject dataWsdlObject){
        QName QNameString = new QName(NS_XSD, "string");

        String temp="";
        Outlier outlier= null;

        String methodName = ".getDataRunOutliers()";

        String dataEndPoint = dataWsdlObject.getEndPoint();
        String dataQNameService = dataWsdlObject.getService();
        String dataQNamePort = dataWsdlObject.getPort();
        String dataNameSpace = dataWsdlObject.getNameSpace();

        String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
        String algorithmQNameService = algorithmWsdlObject.getService();
        String algorithmQNamePort = algorithmWsdlObject.getPort();
        String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

        ServiceFactory factory = null;
        try {
            factory = ServiceFactory.newInstance();
        } catch (ServiceException e) {
            System.out.println(CLASSNAME + methodName + "caught
                      ServiceException for Service " + dataQNameService +" : "
                      + e.toString());
        }

        String ns = BODY_NAMESPACE_VALUE;
        if(algorithmNameSpace != null){
            ns = algorithmNameSpace;
        }

        try {
```

```
            Service service = factory.createService(
                                    new QName(algorithmQNameService));
            QName port = new QName(algorithmQNamePort);

            Call callOutlier = service.createCall(port);
            callOutlier.setTargetEndpointAddress(algorithmEndPoint);

            callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
                new Boolean(true));
            callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            callOutlier.setReturnType(QNameString);
            callOutlier.setOperationName(new QName(ns,
                "getDataRunOutliers"));
            callOutlier.addParameter("String_1",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_2",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_3",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_4",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_5",QNameString, ParameterMode.IN);
            callOutlier.addParameter("String_6",QNameString, ParameterMode.IN);

            String[] params = {  dataQNameService, dataQNamePort,
                                 dataEndPoint, dataNameSpace,
                                 String.valueOf(combinedMean),
                                 String.valueOf(combinedVariance)};

            temp = (String)callOutlier.invoke(params);
            outlier = new Outlier();

            String str1[] = new String[3];
            StringTokenizer stoken1 = new StringTokenizer(temp, "|");
            int i1=0;
            while(stoken1.hasMoreElements())
                str1[i1++] = stoken1.nextToken();
                outlier.setDataSet(str1[0]);
                outlier.setRange(str1[1]);
                outlier.setOutliers(str1[2]);
        } catch (ServiceException e) {
            System.out.println( CLASSNAME + methodName + "caught
                 ServiceException " + dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println(CLASSNAME + methodName + "caught
                 ServiceException "+ dataQNameService +" : " + e.toString());
        } catch(Exception e){
            e.printStackTrace();
        }
       return outlier;
    }
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: WebClient.java
* Purpose: Main Class in Web Interface. Gets the Registries, Runs
*          the web service and saves the results for JSPs to display

*/
package webclient;

import javax.xml.namespace.QName;
import javax.xml.rpc.*;
import java.io.IOException;
import java.rmi.RemoteException;
import java.util.List;
import java.util.StringTokenizer;

public class WebClient {

    private String BODY_NAMESPACE_VALUE = "urn:Foo";
    private String ENCODING_STYLE_PROPERTY =
                        "javax.xml.rpc.encodingstyle.namespace.uri";
    private String NS_XSD = "http://www.w3.org/2001/XMLSchema";
    private String URI_ENCODING =
                        "http://schemas.xmlsoap.org/soap/encoding/";
    private String CLASSNAME = "webclient.WebClient";

    private ResultMeanVariance resultMeanVariance;
    public ResultMeanVariance getResultMeanVariance() {
        return resultMeanVariance;
    }
    public void setResultMeanVariance(ResultMeanVariance resultMeanVariance) {
        this.resultMeanVariance = resultMeanVariance;
    }

    private List resultMeanVarianceList;

    public List getResultMeanVarianceList() {
        return resultMeanVarianceList;
    }
    public void setResultMeanVarianceList(List resultMeanVarianceList) {
        this.resultMeanVarianceList = resultMeanVarianceList;
    }

    private List outlierList;

    public WebClient() {
    }

    public List getOutlierList() {
        return outlierList;
    }
    public void setOutlierList(List outlierList) {
        this.outlierList = outlierList;
    }

    private double combinedMean =0;
    private double combinedVariance =0;
```

```
public double getCombinedMean() {
    return combinedMean;
}
public void setCombinedMean(double combinedMean) {
    this.combinedMean = combinedMean;
}

public double getCombinedVariance() {
    return combinedVariance;
}
public void setCombinedVariance(double combinedVariance) {
    this.combinedVariance = combinedVariance;
}

public void getCombinedMeanVariance(List resultMeanVarianceList){
   int count1,count2;
   double mean1,variance1,mean2,variance2;

    if(resultMeanVarianceList.size()>1){
        ResultMeanVariance resultMeanVariance =
              (ResultMeanVariance) resultMeanVarianceList.get(0);
        count1 = resultMeanVariance.getNoOfElements();
        mean1 = resultMeanVariance.getMean();
        variance1 = resultMeanVariance.getVariance();

        resultMeanVariance =
              (ResultMeanVariance) resultMeanVarianceList.get(1);
        count2 = resultMeanVariance.getNoOfElements();
        mean2 = resultMeanVariance.getMean();
        variance2 = resultMeanVariance.getVariance();

        combinedMeanVariance(mean1, variance1, count1,
                mean2, variance2, count2);

    }


}

private void combinedMeanVariance(double dmean1,double dvar1, int ct1,
                                  double dmean2,double dvar2, int ct2) {
    double cmean = 0, cvar = 0, cvarsqr = 0;

    cmean = (((dmean1 * ct1) + (dmean2 * ct2))/(ct1 + ct2));
    cvarsqr = (((dvar1*dvar1)/(ct1)) + ( (dvar2 * dvar2)/(ct2)))*(ct1+ct2);
    cvar=Math.sqrt(cvarsqr);

    combinedMean = cmean;
    combinedVariance = cvar;
}


public ResultMeanVariance getMeanVarianceForData(WSDLObject dataWsdlObject,
                                        WSDLObject algorithmWsdlObject){

    String methodName=".getMeanVarianceForData()";
    ResultMeanVariance resultMeanVariance = null;

    try {
        String output1 = callMeanVarianceForData(dataWsdlObject,
                                            algorithmWsdlObject);
        System.out.println(CLASSNAME + methodName + " : " + output1);
        String str1[] = new String[3];
```

```
        StringTokenizer stoken1 = new StringTokenizer(output1, ",");
        int i1=0;
        while(stoken1.hasMoreElements())
            str1[i1++] = stoken1.nextToken();
            resultMeanVariance = new ResultMeanVariance();
            Double I1 = new Double(str1[0]);
            resultMeanVariance.setNoOfElements(I1.intValue());
            Double dm1 = new Double(str1[1]);
            resultMeanVariance.setMean(dm1.doubleValue());
            Double dv1 = new Double(str1[2]);
            resultMeanVariance.setVariance(dv1.doubleValue());
    }catch(Exception e){
        e.printStackTrace();
    }
    return resultMeanVariance;
   }


public Outlier getOutliersForData(WSDLObject dataWsdlObject,
                                  WSDLObject algorithmWsdlObject,
                                  double cmean,
                                  double cvar){
    String output = null;
    Outlier outlier = new Outlier();
    try {
        output = callOutliersForData(dataWsdlObject, algorithmWsdlObject,
                                     cmean, cvar);

        String str1[] = new String[3];
        StringTokenizer tokens = new StringTokenizer(output, "|");
        int i1=0;
        while(tokens.hasMoreElements())
            str1[i1++] = tokens.nextToken();
            outlier.setDataSet(str1[0]);
            outlier.setRange(str1[1]);
            outlier.setOutliers(str1[2]);
    } catch (Exception e) {
        System.out.println("Exception in
                webclient.Client.getOutliersForData() : " + e.toString());
    }
    return outlier;
}

public String callMeanVarianceForData(WSDLObject dataWsdlObject,
                                      WSDLObject algorithmWsdlObject){

    QName QNameString = new QName(NS_XSD, "string");
    String methodName = ".callMeanVarianceForData()";
    String temp ="";

    String dataEndPoint = dataWsdlObject.getEndPoint();
    String dataQNameService = dataWsdlObject.getService();
    String dataQNamePort = dataWsdlObject.getPort();
    String dataNameSpace = dataWsdlObject.getNameSpace();

    String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
    String algorithmQNameService = algorithmWsdlObject.getService();
    String algorithmQNamePort = algorithmWsdlObject.getPort();
    String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

    ServiceFactory factory = null;
    try {
        factory = ServiceFactory.newInstance();
```

```
        } catch (ServiceException e) {
            System.out.println(CLASSNAME + methodName + " caught
                  ServiceException for Service " + dataQNameService +" : " +
                   e.toString());
        }

        String ns = BODY_NAMESPACE_VALUE;
        if(dataNameSpace != null){
            ns = dataNameSpace;
        }

        System.out.println("Entered " + CLASSNAME + methodName);
        System.out.println("End Point : " + dataEndPoint);
        try {
            Service service = factory.createService(
                        new QName(dataQNameService));
            QName port = new QName(dataQNamePort);

            Call call = service.createCall(port);
            call.setTargetEndpointAddress(dataEndPoint);

            call.setProperty(Call.SOAPACTION_USE_PROPERTY, new Boolean(true));
            call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            call.setReturnType(QNameString);
            call.setOperationName(new QName(ns, "getMeanVariance"));
            call.addParameter("String_1", QNameString, ParameterMode.IN);
            call.addParameter("String_2", QNameString, ParameterMode.IN);
            call.addParameter("String_3", QNameString, ParameterMode.IN);
            call.addParameter("String_4", QNameString, ParameterMode.IN);
            String[] param = {algorithmEndPoint, algorithmQNameService,
                              algorithmQNamePort, algorithmNameSpace};
            temp = (String) call.invoke(param);
        } catch (ServiceException e) {
            System.out.println("ServiceException in callMeanVarianceForData1
                        for Service " + dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println("RemoteException in callMeanVarianceForData1 " +
                    dataQNameService +" : " + e.toString());
        } catch (IOException e) {
            System.out.println("IOException in callMeanVarianceForData1 " +
                    dataQNameService +" : " + e.toString());
        }
        return temp ;
    }


    public String callOutliersForData(WSDLObject dataWsdlObject,
                                      WSDLObject algorithmWsdlObject,
                                             double combinedMean,
                                             double combinedVariance){
        QName QNameString = new QName(NS_XSD, "string");
        String methodName=".callOutliersForData()";
        String temp="";
        String dataEndPoint = dataWsdlObject.getEndPoint();
        String dataQNameService = dataWsdlObject.getService();
        String dataQNamePort = dataWsdlObject.getPort();
        String dataNameSpace = dataWsdlObject.getNameSpace();

        String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
        String algorithmQNameService = algorithmWsdlObject.getService();
        String algorithmQNamePort = algorithmWsdlObject.getPort();
```

```
        String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

        System.out.println("Entered " + CLASSNAME + methodName);
        System.out.println("Endpoint address = " +  dataEndPoint);

        ServiceFactory factory = null;
        try {
            factory = ServiceFactory.newInstance();
        } catch (ServiceException e) {
            System.out.println(CLASSNAME + methodName+ " caught
                        ServiceException for Service " + dataQNameService
                        +" : " + e.toString());
        }
        String ns = BODY_NAMESPACE_VALUE;
        if(dataNameSpace != null){
            ns = dataNameSpace;
        }

        try {
            Service service = factory.createService(
                            new QName(dataQNameService));
            QName port = new QName(dataQNamePort);

            Call callOutlier = service.createCall(port);
            callOutlier.setTargetEndpointAddress(dataEndPoint);

            callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
                new Boolean(true));
            callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            callOutlier.setReturnType(QNameString);
            callOutlier.setOperationName(new QName(ns,"getOutliers"));
            callOutlier.addParameter("String_1", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_2", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_3", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_4", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_5", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_6", QNameString,ParameterMode.IN);
            String[] params = { algorithmEndPoint, algorithmQNameService,
                            algorithmQNamePort, algorithmNameSpace,
                            String.valueOf(combinedMean),
                            String.valueOf(combinedVariance)};

            temp = (String)callOutlier.invoke(params);
         } catch (ServiceException e) {
            System.out.println("ServiceException in callOutliersForData " +
                    dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println("RemoteException in callOutliersForData " +
                    dataQNameService +" : " + e.toString());
        }
        return temp;
    }

    public ResultMeanVariance getDataRunMeanVariance(
                    WSDLObject algorithmWsdlObject, WSDLObject dataWsdlObject){

        QName QNameString = new QName(NS_XSD, "string");

        String temp="";
        ResultMeanVariance resultMeanVariance = null;
        String methodName = ".getDataRunMeanVariance()";
```

```
String dataEndPoint = dataWsdlObject.getEndPoint();
String dataQNameService = dataWsdlObject.getService();
String dataQNamePort = dataWsdlObject.getPort();
String dataNameSpace = dataWsdlObject.getNameSpace();

String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
String algorithmQNameService = algorithmWsdlObject.getService();
String algorithmQNamePort = algorithmWsdlObject.getPort();
String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

ServiceFactory factory = null;
try {
    factory = ServiceFactory.newInstance();
} catch (ServiceException e) {
    System.out.println(CLASSNAME + methodName + "caught
                       ServiceException for Service " +
                       dataQNameService +" : " + e.toString());
}
String ns = BODY_NAMESPACE_VALUE;
if(algorithmNameSpace != null){
    ns = algorithmNameSpace;
}

try {
    Service service = factory.createService(
                              new QName(algorithmQNameService));
    QName port = new QName(algorithmQNamePort);

    Call callOutlier = service.createCall(port);
    callOutlier.setTargetEndpointAddress(algorithmEndPoint);

    callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
                            new Boolean(true));
    callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
    callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

    callOutlier.setReturnType(QNameString);
    callOutlier.setOperationName(
                         new QName(ns,"getDataRunMeanVariance"));
    callOutlier.addParameter("String_1", QNameString,ParameterMode.IN);
    callOutlier.addParameter("String_2", QNameString,ParameterMode.IN);
    callOutlier.addParameter("String_3", QNameString,ParameterMode.IN);
    callOutlier.addParameter("String_4", QNameString,ParameterMode.IN);

    String[] params = {  dataQNameService, dataQNamePort,
                         dataEndPoint, dataNameSpace};

    temp = (String)callOutlier.invoke(params);

    String str1[] = new String[3];
    StringTokenizer stoken1 = new StringTokenizer(temp, ",");
    int i1=0;
    while(stoken1.hasMoreElements())
        str1[i1++] = stoken1.nextToken();
        resultMeanVariance = new ResultMeanVariance();
        Double I1 = new Double(str1[0]);
        resultMeanVariance.setNoOfElements(I1.intValue());
        Double dm1 = new Double(str1[1]);
        resultMeanVariance.setMean(dm1.doubleValue());

       Double dv1 = new Double(str1[2]);
        resultMeanVariance.setVariance(dv1.doubleValue());
```

```
        } catch (ServiceException e) {
            System.out.println( CLASSNAME + methodName + "caught
                                ServiceException " +
                                dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println(CLASSNAME + methodName + "caught
                                ServiceException "+
                                dataQNameService +" : " + e.toString());
        } catch(Exception e){
            e.printStackTrace();
        }
        return resultMeanVariance;
    }

    public Outlier getDataRunOutliers(WSDLObject algorithmWsdlObject,
                                      WSDLObject dataWsdlObject){
        QName QNameString = new QName(NS_XSD, "string");

        String temp="";
        Outlier outlier= null;

        String methodName = ".getDataRunOutliers()";

        String dataEndPoint = dataWsdlObject.getEndPoint();
        String dataQNameService = dataWsdlObject.getService();
        String dataQNamePort = dataWsdlObject.getPort();
        String dataNameSpace = dataWsdlObject.getNameSpace();

        String algorithmEndPoint = algorithmWsdlObject.getEndPoint();
        String algorithmQNameService = algorithmWsdlObject.getService();
        String algorithmQNamePort = algorithmWsdlObject.getPort();
        String algorithmNameSpace = algorithmWsdlObject.getNameSpace();

        ServiceFactory factory = null;
        try {
            factory = ServiceFactory.newInstance();
        } catch (ServiceException e) {
            System.out.println(CLASSNAME + methodName + "caught
                               ServiceException for Service " +
                               dataQNameService +" : " + e.toString());
        }

        String ns = BODY_NAMESPACE_VALUE;
        if(algorithmNameSpace != null){
            ns = algorithmNameSpace;
        }

        try {
            Service service = factory.createService(
                                    new QName(algorithmQNameService));
            QName port = new QName(algorithmQNamePort);

            Call callOutlier = service.createCall(port);
            callOutlier.setTargetEndpointAddress(algorithmEndPoint);

            callOutlier.setProperty(Call.SOAPACTION_USE_PROPERTY,
                                    new Boolean(true));
            callOutlier.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            callOutlier.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);

            callOutlier.setReturnType(QNameString);
            callOutlier.setOperationName(new
                                         QName(ns, "getDataRunOutliers"));
```

```
            callOutlier.addParameter("String_1", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_2", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_3", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_4", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_5", QNameString,ParameterMode.IN);
            callOutlier.addParameter("String_6", QNameString,ParameterMode.IN);

            String[] params = {  dataQNameService, dataQNamePort,
                                 dataEndPoint, dataNameSpace,
                                 String.valueOf(combinedMean),
                                 String.valueOf(combinedVariance)};

            temp = (String)callOutlier.invoke(params);
            outlier = new Outlier();

            String str1[] = new String[3];
            StringTokenizer stoken1 = new StringTokenizer(temp, "|");
            int i1=0;
            while(stoken1.hasMoreElements())
                str1[i1++] = stoken1.nextToken();
                outlier.setDataSet(str1[0]);
                outlier.setRange(str1[1]);
                outlier.setOutliers(str1[2]);

        } catch (ServiceException e) {
            System.out.println( CLASSNAME + methodName + "caught
                                ServiceException " +
                                dataQNameService +" : " + e.toString());
        } catch (RemoteException e) {
            System.out.println(CLASSNAME + methodName + "caught
                               ServiceException "+
                               dataQNameService +" : " + e.toString());
        } catch(Exception e){
            e.printStackTrace();
        }
       return outlier;
    }
}
```

```
/**
*
* Vivek Mongolu        Master's Thesis                    Fall 2004
*
* File: RunWebService.java
* Purpose:This class actuallys runs the web services.Its used by the web client
*/

package webclient;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.List;
import java.util.LinkedList;
import java.util.Date;

public class RunWebService {
    private static String CLASSNAME="webclient.RunWebService";

    public RunWebService() {
    }

    private String choice;
    private WSDLObject mvAlgoWsdlObject = null;
    private WSDLObject outlierAlgoWsdlObject = null;

    private List algoToDataMeanVarianceList = null;
    private List dataToAlgoMeanVarianceList = null;
    private List algoToDataOutlierList = null;
    private List dataToAlgoOutlierList = null;

    private Date algoToDataStartTime = null;
    private Date algoToDataEndTime = null;
    private Date dataToAlgoStartTime = null;
    private Date dataToAlgoEndTime = null;

    public List getAlgoToDataMeanVarianceList() {
        return algoToDataMeanVarianceList;
    }

    public void setAlgoToDataMeanVarianceList(List algoToDataMeanVarianceList)
    {
        this.algoToDataMeanVarianceList = algoToDataMeanVarianceList;
    }

    public List getDataToAlgoMeanVarianceList() {
        return dataToAlgoMeanVarianceList;
    }

    public void setDataToAlgoMeanVarianceList(List dataToAlgoMeanVarianceList)
    {
        this.dataToAlgoMeanVarianceList = dataToAlgoMeanVarianceList;
    }

    public List getAlgoToDataOutlierList() {
        return algoToDataOutlierList;
    }

    public void setAlgoToDataOutlierList(List algoToDataOutlierList) {
        this.algoToDataOutlierList = algoToDataOutlierList;
```

```
}

public List getDataToAlgoOutlierList() {
    return dataToAlgoOutlierList;
}

public void setDataToAlgoOutlierList(List dataToAlgoOutlierList) {
    this.dataToAlgoOutlierList = dataToAlgoOutlierList;
}

public Date getAlgoToDataStartTime() {
    return algoToDataStartTime;
}

public void setAlgoToDataStartTime(Date algoToDataStartTime) {
    this.algoToDataStartTime = algoToDataStartTime;
}

public Date getAlgoToDataEndTime() {
    return algoToDataEndTime;
}

public void setAlgoToDataEndTime(Date algoToDataEndTime) {
    this.algoToDataEndTime = algoToDataEndTime;
}

public Date getDataToAlgoStartTime() {
    return dataToAlgoStartTime;
}

public void setDataToAlgoStartTime(Date dataToAlgoStartTime) {
    this.dataToAlgoStartTime = dataToAlgoStartTime;
}

public Date getDataToAlgoEndTime() {
    return dataToAlgoEndTime;
}

public void setDataToAlgoEndTime(Date dataToAlgoEndTime) {
    this.dataToAlgoEndTime = dataToAlgoEndTime;
}

public String getChoice() {
    return choice;
}

public void setChoice(String choice) {
    this.choice = choice;
}

public void run(HttpServletRequest request){
    String methodName=".run()";
    WSDLParser wsdlParser = new WSDLParser();
    HttpSession session = request.getSession();

    String meanVarianceWSDLFile =
           (String) session.getAttribute("meanVarianceWSDLFile");
    String outlierWSDLFile =
           (String) session.getAttribute("outlierWSDLFile");

    if(meanVarianceWSDLFile != null &&
            meanVarianceWSDLFile.trim().length() > 0){
        System.out.println("Creating mvAlgoWsdlObject ");
```

```
            wsdlParser.parseWSDLFile(meanVarianceWSDLFile);
            mvAlgoWsdlObject = wsdlParser.getWsdlObject();
        }

        if(outlierWSDLFile != null && outlierWSDLFile.trim().length() > 0){
            System.out.println("Creating outlierAlgoWsdlObject ");
            wsdlParser.parseWSDLFile(outlierWSDLFile);
            outlierAlgoWsdlObject = wsdlParser.getWsdlObject();
        }

        List dataMap = new LinkedList();

        List dataOrgs = (List) session.getAttribute("dataOrgs");
        System.out.println("Creating dataMap");
        for (int i = 0; i < dataOrgs.size(); i++) {
            JAXRQueryResult jaxrQueryResult =
                            (JAXRQueryResult) dataOrgs.get(i);
            String dataWSDLFile =
                    jaxrQueryResult.getServiceBindingURI()+"?WSDL";

            // parse the WSDL files and save the wsdlobject in a List
            wsdlParser.parseWSDLFile(dataWSDLFile);
            dataMap.add(wsdlParser.getWsdlObject());
        }
        System.out.println("Done creating dataMap");
        WebClient client = new WebClient();
        System.out.println("Created client Object");
        if(choice.equalsIgnoreCase("1")){
            // Algorithm to Data
            algoToDataStartTime = new Date();
            runAlgoToData(client, dataMap);
            algoToDataEndTime = new Date();
        }else if(choice.equalsIgnoreCase("2")){
            // Data to Algorithm
            dataToAlgoStartTime = new Date();
            runDataToAlgo(client, dataMap);
            dataToAlgoEndTime = new Date();
        }else {
            // Both
            algoToDataStartTime = new Date();
            runAlgoToData(client, dataMap);
            algoToDataEndTime = new Date();
            dataToAlgoStartTime = new Date();
            runDataToAlgo(client, dataMap);
            dataToAlgoEndTime = new Date();
        }
    }

    public void runAlgoToData(WebClient client, List dataMap){
        String methodName = ".runAlgoToData()";
        System.out.println("Entered " + CLASSNAME + methodName);

        algoToDataMeanVarianceList = new LinkedList();
        algoToDataOutlierList = new LinkedList();

        System.out.println("Algorithm to Data");
        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running MeanVariance Web Service on Data " +
                               (i+1));
            ResultMeanVariance resultMeanVariance =
               client.getMeanVarianceForData(dataWSDLObject, mvAlgoWsdlObject);
            algoToDataMeanVarianceList.add(resultMeanVariance);
```

```
            System.out.println(resultMeanVariance);
        }

        if(algoToDataMeanVarianceList.size() > 1){
            System.out.println("Calculating CombinedMean and
                                CombinedVariance");
            client.getCombinedMeanVariance(algoToDataMeanVarianceList);
        }
        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running Outlier Web Service on Data " + (i+1));
            Outlier outlier = client.getOutliersForData( dataWSDLObject,
                                                outlierAlgoWsdlObject,
                                            client.getCombinedMean(),
                                        client.getCombinedVariance());
            algoToDataOutlierList.add(outlier);
            System.out.println(outlier);
        }
    }

    public void runDataToAlgo(WebClient client, List dataMap){
        String methodName = ".runDataToAlgo()";
        System.out.println("Entered " + CLASSNAME+methodName);

        dataToAlgoMeanVarianceList = new LinkedList();
        dataToAlgoOutlierList = new LinkedList();

        System.out.println("Data to Algorithm");
        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running " + dataWSDLObject.getService() +
                               " on MeanVariance Web Service");
            ResultMeanVariance resultMeanVariance =
                client.getDataRunMeanVariance(mvAlgoWsdlObject,dataWSDLObject);
            dataToAlgoMeanVarianceList.add(resultMeanVariance);
            System.out.println(resultMeanVariance);
        }

        if(dataToAlgoMeanVarianceList.size() > 1){
            System.out.println("Calculating CombinedMean and
                                CombinedVariance");
            client.getCombinedMeanVariance(dataToAlgoMeanVarianceList);
        }

        for (int i = 0; i < dataMap.size(); i++) {
            WSDLObject dataWSDLObject = (WSDLObject) dataMap.get(i);
            System.out.println("Running " + dataWSDLObject.getService() +
                               " on Outlier Web Service");
            Outlier outlier = client.getDataRunOutliers(outlierAlgoWsdlObject,
                                                        dataWSDLObject);
            dataToAlgoOutlierList.add(outlier);
            System.out.println(outlier);
        }
    }
}
```

Index.jsp

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: index.jsp
* Purpose: main page in client. Displays all the registries available
*/

<%@ page import="webclient.Registries, java.util.List, webclient.Registry"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<html>
    <head>
        <title>Thesis - Web Service</title>
        <link href="stylesheet.css" rel="stylesheet" type="text/css">
    </head>

    <jsp:useBean id="registries" class="webclient.Registries"/>

    <body bgcolor="white">
        <div class="headingBox">
            <center>
                Distributed Data Mining using Web Services
            </center>
        </div>
        <br/><br/>
        <form action="display_registry_services.jsp" method="post">
            <table width="100%" align="center">
                <tr><td>  </td></tr>
                <tr>
                    <td class="largeTextBlue" >
                        Select a Registry to search for Algorithm and Data Web
                        Services
                    </td>
                </tr>
                <tr><td>  </td></tr>
                <tr>
                    <td align="center">
                        <select name="registry" class="mediumTextBlue">
                            <c:forEach items="${registries.registriesFromDB}"
                                       var="registry" >
                                <option value="<c:out value="${registry.url}"
                                        escapeXml="true" />">
                                    <c:out value="${registry.name}"/>
                                </option>
                            </c:forEach>
                        </select>
                    </td>
                </tr>
                <tr><td> </td></tr>
                <tr>
                    <td align="center">
                        <input type="submit" value="Submit"/>
                    </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

display_registry_services.jsp

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: display_registry_services.jsp
* Purpose: Displays all the Algorithm and Data webservices registered at a
*          selected Registry Server
*/

<%@ page import="webclient.JAXRQuery, java.util.Collection,java.util.List,
                 webclient.JAXRQueryResult"%>
<%@ page contentType="text/html" %>

<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<%
    JAXRQuery jaxrQuery = new JAXRQuery();

    String meanVarianceWSDLFile = null;
    String outlierWSDLFile = null;

    String registryURL = request.getParameter("registry");
    String algorithmString = "algorithmwebservice";
    String dataString = "dataservices";

    List algorithmOrgs = jaxrQuery.getDataMiningServices(algorithmString,
                                                         registryURL);
    session.setAttribute("algorithmOrgs", algorithmOrgs);
    System.out.println("Algorithm Organizations : " + algorithmOrgs.size());

    List dataOrgs = jaxrQuery.getDataMiningServices(dataString, registryURL);
    session.setAttribute("dataOrgs", dataOrgs);
    System.out.println("Data Organizations : " + dataOrgs.size());

    for (int i = 0; i < algorithmOrgs.size(); i++) {
       JAXRQueryResult jaxrQueryResult = (JAXRQueryResult) algorithmOrgs.get(i);
       if(jaxrQueryResult.getServiceDescription().indexOf("Mean") != -1){
            meanVarianceWSDLFile =
                     jaxrQueryResult.getServiceBindingURI()+"?WSDL";
            session.setAttribute("meanVarianceWSDLFile", meanVarianceWSDLFile);
       }else{
            outlierWSDLFile = jaxrQueryResult.getServiceBindingURI()+"?WSDL";
            session.setAttribute("outlierWSDLFile", outlierWSDLFile);
       }
    }
%>

<html>
    <head>
        <title>Thesis - Web Service</title>
        <link href="stylesheet.css" rel="stylesheet" type="text/css">
    </head>

    <body bgcolor="white">
        <div class="headingBox">
            <center>
                Distributed Data Mining using Web Services
            </center>
        </div>
        <br/>
```

```
<div class="mediumError">
    <c:forEach  var="message" items="${requestScope.messages}">
        <c:out value="${message}"/><br />
    </c:forEach>
</div>
<br/>
<form action="run_webservices.jsp" method="post">
    <table width="100%" align="center">
        <tr>
            <td class="largeTextBlue" >
                Following Web Services are available at -
                <%=registryURL%>
            </td>
        </tr>
        <tr>
            <td>
                 
            </td>
        </tr>
        <tr>
            <td align="center">
                <table class="contentBox">
                    <tr class="contentBox">
                        <td>
                            <b>Algorithm Web Services</b> <br/><br/>
                             <%
                                JAXRQueryResult queryResult;
                                for (int i = 0;
                                     i < algorithmOrgs.size(); i++) {
                                    queryResult =
                                (JAXRQueryResult)algorithmOrgs.get(i);
                                %>

                               <li><%=queryResult.getServiceDescription()%>
                               <br/><br/>
                                    <%
                                        }
                                    %>
                        </td>
                    </tr>
                    <tr>
                        <td>
                             
                        </td>
                    </tr>
                    <tr class="contentBox">
                        <td>
                            <b>The following Data Web Services are
                               avialable:</b><br/><br/>
                            <%
                                for (int i = 0; i < dataOrgs.size();
                                     i++) {
                                    queryResult =
                                     (JAXRQueryResult) dataOrgs.get(i);
                            %>
                           <li><%=queryResult.getServiceDescription()%>
                            <br/><br/>
                            <%
                                }
                            %>
                        </td>
                    </tr>
                </table>
```

```
                    </td>
                </tr>
                <tr>
                    <td>
                         
                    </td>
                </tr>
                <tr>
                    <td>
                        <table class="contentBox" align="center">
                            <tr class="contentBox" >
                                <td>
                                    <b>Select the Operation you want to
                                       perform</b><br/><br/><br/>
                                     <input type="radio" name="choice"
                                         value="1"> Algorithm to Data<br/><br/>
                                    <input type="radio" name="choice"
                                         value="2"> Data to Algorithm<br/><br/>
                                    <input type="radio" name="choice"
                                         value="3"> Both<br/><br/>
                                </td>
                            </tr>
                            <tr>
                                <td>
                                     
                                </td>
                            </tr>
                            <tr>
                                <td align="center">
                                    <input type="submit" value="Submit">
                                </td>
                            </tr>
                        </table>
                    </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

run_webservices.jsp

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: run_webservices.jsp
* Purpose: Page runs the web services and displays the results, Uses
*          RunWebService Class
*/
<%@page import="java.util.List, webclient.*, java.util.Map,
                java.util.HashMap, java.util.LinkedList" %>

<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<jsp:useBean id="client" class="webclient.WebClient"/>

<jsp:useBean id="runwebservice" class="webclient.RunWebService"/>
<jsp:setProperty name="runwebservice" property="*" />

<html>
    <head>
        <title>Thesis - Web Service</title>
        <link href="stylesheet.css" rel="stylesheet" type="text/css">
    </head>

    <body bgcolor="white">
        <div class="headingBox">
            <center>
                Distributed Data Mining using Web Services
            </center>
        </div>
        <br/><br/>

        <%
            runwebservice.run(request);
            request.setAttribute("runwebservice",runwebservice );
        %>

        <c:choose>
            <c:when test='${runwebservice.choice == "1"}'>
               <jsp:include page="algo_to_data.jsp"/>
            </c:when>
            <c:when test='${runwebservice.choice == "2"}'>
               <jsp:include page="data_to_algo.jsp"/>
            </c:when>
            <c:otherwise>
               <jsp:include page="algo_to_data.jsp"/>
               <br/><br/>
               <jsp:include page="data_to_algo.jsp"/>
            </c:otherwise>
        </c:choose>

    </body>
</html>
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: algo_to_data.jsp
* Purpose: Page displays the result of running Algorithm Web Service on Data
*          Web Service
*/

<%@ page import="webclient.RunWebService"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<%
    RunWebService runWebService =
                (RunWebService)request.getAttribute("runwebservice");
%>
<div class="subHeadingBox">
    <center>
        Algorithm To Data Results
    </center>
</div>
<br/>
<br/>

<table class="contentBox">
    <c:if test="${not empty runwebservice.algoToDataMeanVarianceList}">
        <tr>
            <td>
                Result of Running Mean Variance Web Service on Data Web Service
            </td>
        </tr>
    </c:if>
    <c:forEach items="${runwebservice.algoToDataMeanVarianceList}"
               var="meanVariance">
        <tr>
            <td>
                
            </td>
        </tr>
        <tr>
            <td>
                 No : of Elements :
                  <c:out value="${meanVariance.noOfElements}"/>
            </td>
        </tr>
        <tr>
            <td>
                Mean : <c:out value="${meanVariance.mean}"/>
            </td>
        </tr>
        <tr>
            <td>
                Variance :<c:out value="${meanVariance.variance}"/>
            </td>
        </tr>

    </c:forEach>
</table>
<br/>
<br/>
```

```
<table class="contentBox">
    <c:if test="${not empty runwebservice.algoToDataOutlierList}">
        <tr>
            <td>
                Result of Running Outlier Web Service on Data Web Service
            </td>
        </tr>
    </c:if>
    <c:forEach items="${runwebservice.algoToDataOutlierList}"   var="outlier">

        <tr>
            <td> 
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.dataSet}"/>
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.range}"/>
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.outliers}"/>
            </td>
        </tr>
    </c:forEach>
</table>
<br/><br/>
<table class="contentBox">
    <tr>
        <td> 
        </td>
    </tr>
    <tr>
        <td>
            Start Time : <c:out value="${runwebservice.algoToDataStartTime}"/>
        </td>
    </tr>
    <tr>
        <td> 
        </td>
    </tr>
    <tr>
        <td>
            End Time : <c:out value="${runwebservice.algoToDataEndTime}"/>
        </td>
    </tr>
    <tr>
        <td> 
        </td>
    </tr>
</table>
```

```
/**
*
* Vivek Mongolu        Master's Thesis                          Fall 2004
*
* File: data_to_algo.jsp
* Purpose: Page displays the result of running Data Web Service on Algorithm
*          Web Service
*/

<%@ page import="webclient.RunWebService"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<%
    RunWebService runWebService =
                (RunWebService) request.getAttribute("runwebservice");
%>
<div class="subHeadingBox">
    <center>
        Data to Algorithm Results
    </center>
</div>
<br/>
<br/>
<table class="contentBox">
    <c:if test="${not empty runwebservice.dataToAlgoMeanVarianceList}">
        <tr>
            <td>
                Result of Running Mean Variance Web Service on Data Web Service
            </td>
        </tr>
    </c:if>
    <c:forEach items="${runwebservice.dataToAlgoMeanVarianceList}"
               var="meanVariance">
        <tr>
            <td> 
            </td>
        </tr>
        <tr>
            <td>
                 No : of Elements :
                 <c:out value="${meanVariance.noOfElements}"/>
            </td>
        </tr>
        <tr>
            <td>
                Mean : <c:out value="${meanVariance.mean}"/>
            </td>
        </tr>
        <tr>
            <td>
                Variance :<c:out value="${meanVariance.variance}"/>
            </td>
        </tr>
    </c:forEach>
</table>
<br/>
<br/>
<table class="contentBox">
    <c:if test="${not empty runwebservice.dataToAlgoOutlierList}">
        <tr>
```

```
            <td>
                Result of Running Outlier Web Service on Data Web Service
            </td>
        </tr>
    </c:if>
    <c:forEach items="${runwebservice.dataToAlgoOutlierList}"   var="outlier">

        <tr>
            <td> 
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.dataSet}"/>
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.range}"/>
            </td>
        </tr>
        <tr>
            <td>
                <c:out value="${outlier.outliers}"/>
            </td>
        </tr>
    </c:forEach>
</table>
<br/>
<br/>
<table class="contentBox">
    <tr>
        <td> 
        </td>
    </tr>
    <tr>
        <td>
            Start Time : <c:out value="${runwebservice.dataToAlgoStartTime}"/>
        </td>
    </tr>
    <tr>
        <td> 
        </td>
    </tr>
    <tr>
        <td>
            End Time : <c:out value="${runwebservice.dataToAlgoEndTime}"/>
        </td>
    </tr>
    <tr>
        <td> 
        </td>
    </tr>
</table>
```

CURRICULUM VITAE

NAME:	Vivek Mongolu
ADDRESS:	Apt 2, 1800 South Third Street Louisville, Ky 40208
DOB:	Secunderabad, India – Nov 2, 1977
EDUCATION AND TRAINING:	B.Tech. (Computer Science) Jawaharlal Nehru Technological University Hyderabad, India M.S. (Computer Science) University of Louisville Louisville, Kentucky