Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp711-716)

A Web Service based Statistical Service

YUE-SHAN CHANG, TONG-YING JUANG
Department of Computer Science and Information Engineering
National Taipei University
151, University Road, Sanhsia, Taipei County,
237 TAIWAN, R.O.C.
http://web.ntpu.edu.tw/~juang

Abstract: - In this paper, we will propose a flexible and scalable statistical environment based on the web service and offer a unified XML-based programming interface for statistical computing. We separate various statistical functionalities into groups of services and design its interface and architecture. All of services are defined as an agent and implemented using web service technology. And all the statistical operations, data, and results are formulated as a XML document. Finally, a prototype was designed and implemented to demonstrate the feasibility of the proposed framework. Statisticians can easily program their solution and the programs can be Write-Once and Run-Anywhere (WORA).

Keywords: - Web Service, Web computing, Statistical system, Statistical computing, Unified Programming Interface.

1 Introduction

1.1 Problems and motivations

As every knows there have some popular statistical software on computer, such as SPSS, SAS, Matlab, Mathematica, and Excel etc. These commercial tools either might be too expensive to offer to general users or have not support programming interface for coding specific model in certain applications. Besides, the proprietary interface results in the computing that could not be executed at anytime and anywhere.

With the rapidly advance of Internet and Information Technology, it is with the trend to offer a web-based statistical computing environment. Many statistical systems have been proposed and built on the WWW for supporting unified user interface and running at anytime and anywhere, for example [1-3][7-8]. These systems offer users a unified user interface on the WWW to execute statistical functionalities in step-by-step and on a dedicated server. Nevertheless, it is not so easy to process some complicated statistical computing in a simple way.

For complicated computing, the R project [9] and S-PLUS statistical library [13] offers programmer a complete set of statistical and numerical library for modeling and programming the solution of their problem domain. In addition, Junji Nakano [10-11] proposes the JASP (JAva based Statistical Processor) - a Statistical package. JASP is based on the client-server model, and use Java RMI (Remote Method Invocation) and distributed computing technology to implement the statistical package. Its client UI (User Interface) applet can use only the server on the Web server computer. The server to which the client connects firstly is called as a main server, and let a main server communicate with other remote servers. SAS/Connect module [16] supports six services for interoperating of statistical computing on a distributed and homogeneous environment. It provides users and applications developers the ability to manage, access, and process data in a distributed environment.

1.2 Objectives

In the web computing, XML proposed by W3C [6] is a well-known and popular Internet standard for information exchange. Due to its simplicity and extensibility, many information platforms and information retrieval environments have applied XML as the data representation and exchange model. These systems use XML as the language for specification, description, programming, or presentation in the applications. Particularly, it is an important industrial standard for information interchange in a heterogeneous environment, which needs a standard format for data exchanging for interoperability, so that the exchanged information can be read, understood, processed, and interchanged easily. XML has been used as one of output format on some statistical software, such as S-PLUS statistical library [13], for further information exchange in heterogeneous environment.

In this paper, we will propose a flexible and scalable statistical environment called MASS (stand

for Multi/web-Agent Statistical System) based on web service computing and offer a unified XML-based programming interface for statistical computing. We design the interface and architecture. The system separates various statistical functionalities into groups of services. All of services in the environment are defined as an agent, implemented using web service technology, and run on distributed environment. And all the statistical operations are formulated as a XML document. Users can ask for their statistical computing by submitting the requests and getting the reply in a XML form over the WWW. The statistical data can be exchanged over heterogeneous environment. And statisticians can easily program their solution and the programs can be Write-Once and Run-Anywhere (WORA) naturally.

This paper is organized as follows. Section 2 mainly depicts the design of the system. We examine some design and implementation issues including the consideration, architecture, computing supports, and implementation of agents etc. Section 3 shows example to demonstrate the environment. Section 4 gives a conclusion.

2 System architecture

2.1 System design consideration

The key features of a distributed computing environment that can be run everywhere are cross-platform. Even current distributed object-oriented programming languages, such as CORBA, JAVA/RMI and Microsoft .NET framework have with the features, most of the statisticians are not familiar with that to programming their solution.

The system implementation is based on the web service and communicates with other agents by sending SOAP [4] message to invoking remote services. The reason of defining a new interface for statistical computing based on XML instead of adopting conventional SOAP semantic of web service is that the SOAP is originally designed to invoke remote object on the web, which has complete and complex syntax for general programmers to coding their programs. This syntax somehow is too complex than general programming language to learn and use for statisticians. SOAP might be very simple when it's narrowed down to a particular application. However, it has not the ability to supply several computation requests and several data sets in a single message, and the freedom to combine requests and data sets arbitrarily. The obvious benefit is the elimination of transfer of results to the user and back to the web service for subsequent processing. It would definitely be something that SOAP can't do, although it's quite easy to implement in XML given its hierarchical nature.

This is the reason why we use the XML that is the standard format and the web service to implement the statistical system on the Internet and the WWW. The XML document is a flat and tag-based document, it is easy to understand and use for programming the solution. Besides, the web services are executed on the WWW that is full transparency for user. So that statisticians can code and run statistical computing anywhere and anytime without the knowledge of any complicated programming language and the platform and the location of remote server.

In the prototype, we group these technologies into a variety of services. A service will be implemented as an agent, as shown in Fig. 1. The service agents (stand for SA) can be classified as *Basic Statistical Agent* (BSA), *Testing Agent* (TA), and *Analysis Agent* (AA) respectively. The BSA consists of functions of *Descriptive Statistics*, *Probability Distribution*, and *Random Number Generation*. The TA includes the functions of *Hypothesis Testing*, *Analysis of Variance*, *Categorical and Discrete Data*, *Nonparametric Statistics* and *Tests of Goodness of Fit*. And the AA has the functions of *Regression Analysis*, *Time Series Analysis*, and *Multivariate Analysis*. Such classification will not only raise the merit of load balance, but also improve the availability. In other words, the architecture will improve the situation of which a dedicated server arise the bottleneck of the system. And a certain server crashed will keep on working of the function in other servers without any effect.

Except for the SAs mentioned above, MASS still have four agents for assisting the user to submit their statistical operations. These are *User Agent* (UA), *Data Agent* (DA), *Dispatching Agent* (DPA), and *Meta Agent* (MA) respectively. The UA provides the user interface and collects the user's data and operations. The DA is responsible for translating user's submission into XML format and forwards the formatted request to DPA. The DA also can consult the MA the formation of specific operations supported by SAs through send the Inquiry request before submitting request to DPA. The DPA will decide target SA of the request and get response from the SA. In the design, user also can send the formatted request to DPA through the UA. The MA is responsible for providing the information of system. Details are depicted in Section 2.4.

Although the architecture and the design can give many advantages in a statistical computing system, it may also arise much overhead due to many transmissions over the network, especially based on the web service computing. Hence, reducing possible communication overhead between client and server is another issue in the system design. We will support multiple operations in a request to a server. The detail is presented in the Section 2.3.

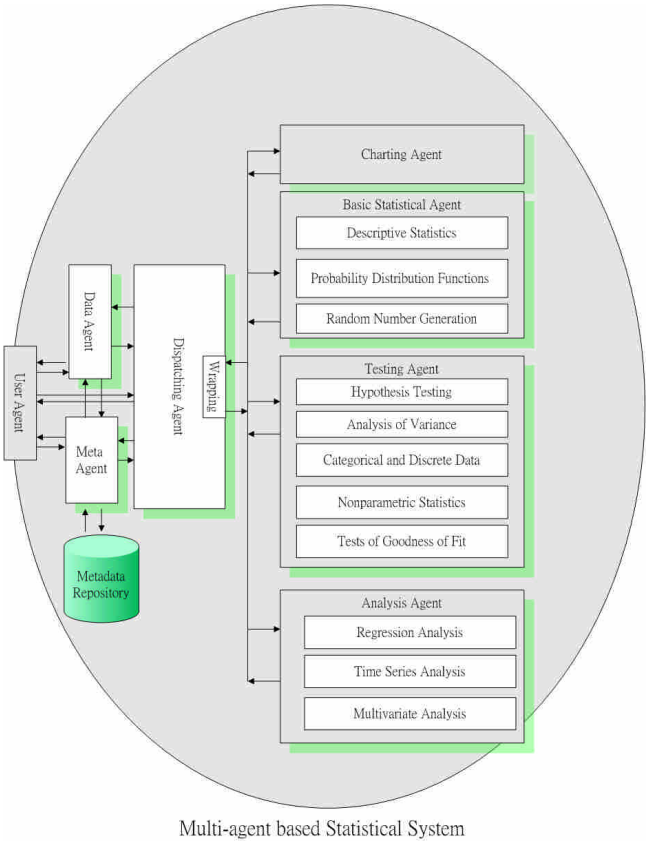

Fig. 1. The system architecture

2.2 Programming interface

In the environment, user sends an XML file to remote SA and gets corresponding responses according to the requested operations. The section will describe the XML format.

Before depicting the XML format, we first introduce the DTD (Document Type Definition) that is used to describe the schema and define the content of XML file. Fig. 2 shows the DTD of request interface. The interface mainly consists of *<USERID>*, *<AGENTLIST>*, *<FUNCTLIST>*, *<OPLIST>*, and *<PARALIST>*. The *<USERID>* identifies the user who submits the request. In the future we can use the tag to judge the user authority and authorized service. The *<AGENTLIST>* lists the source agent if have and target agent who serves the request. The tag can help the target agent to identify the agent to response. The *<FUNCTLIST>* lists the functions. It is because that is possible of more one function in an agent. The *<OPLIST>* lists the operations of which client request. And the *<PARALIST>* lists all the parameter or data set that will be processed by the operations. Among them, the *<FUNCTLIST>*, *<OPLIST>*, and *<PARALIST>* need at least one. In the *<OPLIST>*, there have at least one *<op>* that represents the operation indicated by *<OPNAME>*. The *<INPARA>* indicates the parameter want to be processed.

In addition, the *<PARA>* in the *<PARALIST>* represents the parameter or the data set; user needs to designate the name (*<PARANAME>*), the type (*<TYPE>*) and the content (*<DATA>*) of parameter. The name of operations and parameters can be arranged in any sequence in a request in order to keeping the flexibility. When the data set is a two-dimension array, the *<DATA>* tag can represent a one-dimension array and multiple *<DATA>* tags can represent a two-dimension array.

```
<?xml version = "1.0" encoding = "Big5"?>
<!DOCTYPE MASS_SUBMIT[
<!ELEMENT MASS_SUBMIT (USERID?, AGENTLIST,
FUNCTLIST, OPLIST, PARALIST)>
<! ELEMENT USERID (#PCDATA)>
<! ELEMENT AGENTLIST (FROMAGENT?,
TOAGENT)>
<! ELEMENT FROMAGENT (#PCDATA)>
<! ELEMENT TOAGENT (#PCDATA)>
<! ELEMENT FUNCTLIST (FUNCTION+)>
<! ELEMENT FUNCTION (#PCDATA)>
<! ELEMENT OPLIST (OP+)>
<! ELEMENT OP (OPNAME, INPARA+)>
<! ELEMENT OPNAME (#PCDATA)>
<! ELEMENT INPARA (#PCDATA)>
<! ELEMENT PARALIST (PARA+)>
<! ELEMENT PARA (PARANAME, TYPE, DATA+)>
<! ELEMENT PARANAME (#PCDATA)>
<! ELEMENT TYPE (#PCDATA)>
<! ELEMENT DATA (VALUE+)>
<! ELEMENT VALUE (#PCDATA)>
]>
```

Fig. 2. The request definition DTD

Fig. 3 shows a simple example to demonstrate the use of XML-based interface. The example shows that the request is sent from DPA to *Basic Statistical agent* and uses the *Descriptive Statistics* (DS) function in *Basic Statistical agent*. The operations are to get the *Minimum*, *Maximum*, *Mean*, *Variance* and so on. And the data sets of these operations are *data2*, *data1*, *data2*, and *data3* respectively. The XML file will be sent to remote SA (BSA) that will execute the request and response the result.

Fig. 4 shows the DTD of result. The result represented in XML format may have some merits that will be highlighted in Section 3.7. The semantics are same as request interface except for the *<RESULTLIST>*. There are some *<RESULT>* in the *<RESULTLIST>* that list the result of operations submitted in previous request. Each *<RESULT>* consists of *<OP>*, *<TYPEOFVALUE>*, and *<DATA>*.

```
<?xml version = "1.0" encoding="Big5"?>
<MASS_SUBMIT>
<USERID>guest</USERID>
<AGENTLIST>
<FROMAGENT>Dispatching</FROMAGENT>
<TOAGENT>Basic_Statistical</TOAGENT>
</AGENTLIST>
```

```
<FUNCTLIST>
<FUNCTION>DS</FUNCTION>
</FUNCTLIST>
<OPLIST>
        <OP><OPNAME>Minimum</OPNAME><INPARA>
            data2</INPARA></OP>
        <OP><OPNAME>Maximum</OPNAME><INPARA>
            data1</INPARA></OP>
        <OP><OPNAME>Mean</OPNAME><INPARA>data2
            </INPARA></OP>
        <OP><OPNAME>Variance</OPNAME><INPARA>
            data3</INPARA></OP>
</OPLIST>
        <PARALIST>
        <PARA><PARANAME>data1</PARANAME>
              <TYPE>ArrayOfDouble</TYPE>
        <DATA><VALUE>124,245,216,632,148,97</VALUE>
              </DATA></PARA>
        <PARA><PARANAME>data2</PARANAME>
              <TYPE>ArrayOfDouble</TYPE>
        <DATA><VALUE>4,5,7,9</VALUE></DATA>
              </PARA>
        <PARA><PARANAME>data3</PARANAME>
              <TYPE>ArrayOfDouble</TYPE>
        <DATA><VALUE>1,2,3,4</VALUE></DATA>
              </PARA>
        </PARALIST>
</MASS_SUBMIT>
```

Fig. 3. Submission example

```
<?xml version = "1.0" encoding = "Big5"?>
<!DOCTYPE MASS_RESPONSE[
<!ELEMENT MASS_RESPONSE (USERID?,
AGENTLIST, RESULTLIST)>
<!ELEMENT USERID (#PCDATA)>
<!ELEMENT AGENTLIST (FROMAGENT,
TOAGENT?)>
<!ELEMENT FROMAGENT (#PCDATA)>
<!ELEMENT TOAGENT (#PCDATA)>
<!ELEMENT RESULTLIST (RESULT+)>
<!ELEMENT RESULT (OP, TYPEOFVALUE, DATA)>
<!ELEMENT OP (#PCDATA)>
<!ELEMENT TYPEOFVALUE (#PCDATA)>
<!ELEMENT DATA (VALUE+)>
<!ELEMENT VALUE (#PCDATA)>
]>
```

Fig.4. The response definition DTD

The type of computing result will be distinguishable according to the requested operation. The interface in the request and the reply has a tag to identify the type of parameter. The ArrayOfDouble is used to represent a one-dimension array of double in the parameter while the *MatrixofDouble* representing a two-dimension array of double. In the MASS, a *<VALUE>* tags can show an N x 1 array. For example, <VALUE> a_1, a_2, a_3, …, a_n</VALUE>. The *MatrixofDouble* can use multiple *<VALUE>* tags to represent second dimension. Intuitively, the result returned from remote service can also be post-processed using general programming.

2.3 Computing Support

The interface offers user sending their statistical request to certain agent. The request may involve one or many operations. According to the number of dataset and operation, the computing, supported by the system, can be classified into four types that are: Single Dataset Single Operation (SDSO), Single Dataset Multiple Operations (SDMO), Multiple Datasets Single Operation (MDSO), and Multiple Datasets Multiple Operations (MDMO) respectively. This classification is based on the association of the number of operation and dataset. If one operation just operates on one dataset, we call that SDSO. On the contrary, if multiple operations operate on same dataset, it is called SDMO. Fig. 3 shows the example of SDMO because *Minimum* and *Mean* operate on same dataset (data2). Similarly, one operation can operate on different dataset in a request.

Client is able of merging many operations dispersed on different SAs into a request file if necessary. The request can be sent to the SAs and gets individually the response and merges them for client. The system and the interface obviously offer statisticians a flexible statistical computing environment.

2.4 Service Agent

SA is mainly executing the request submitted by user. All SAs need a unified interface that can help user to send the request and system developer to build system agents easily. So that user can send request to SA and don't take account of the difference of interface among SAs. The diversity of the three SAs is the name of SA and the internal implementation. In other words, *BSAgent*, *AAgent*, and *TAgent* have same invocation interface except for the agent name. In addition, the behind statistical libraries used in the agent are the JMSL [14] announced by Visual Numerics. The JMSL provides many primitive objects and methods of mathematics and statistics in JAVA. An operation in SAs may be to cascade some primitive methods execution.

2.5 Dispatching Service

So far we define a XML programming interface for statisticians for coding their request that can be sent to remote SA and getting associated reply from agents. This section will present the implementation of dispatching program that user can send the request to and getting the reply from SAs.

Deitel [12] lists some systems and architectures for implementing web services, such as Axis, CapeConnect, GULE, IONA Orbix's XMLBus, WASP Lite etc. These systems can be used to implement web services. In our system, we adopt Axis to implement the SAs. But, as mentioned above, user just need to edit simple XML file instead of coding complicated program to use the services; then send out the file to the SAs. These services are transparent to user who doesn't need to know where

the SA located. The dispatching program can decide which agent will be located and where to be sent according to the assigned function in the XML file. The SAs are implemented using web service that the communication between client and server are based on the SOAP message, the dispatching program can be implemented in any programming language. In order to support the future of WORA, we implement a java-based dispatching program to dispatch the XML-based request file.
As mentioned above, the interface of a web service can be represented in WSDL [5]. The programming of dispatching program can export the WSDL file to generate the stub of client program of web service and use the stub to invoke the remote services.

2.6 User Interface

User Agent implements the user interface (UI) and connects to other agents in the MASS. For the reason of user friendly, the user interface is using web browser and ties user agent implemented by Java Server Page. User can submit their request by UI by select operations and input data or by submit a XML file edited previously. Fig. 5 shows the user interface that list the function of AA. Users can select the function and operation, and then input the data.

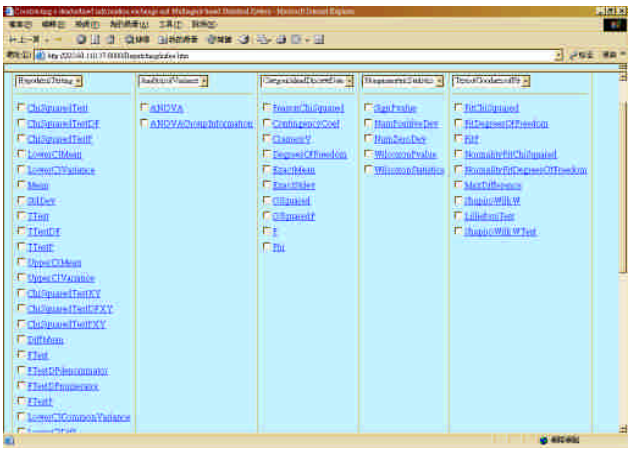

Fig. 5. User interface

The computing result of agent is also represented as XML document. The reasons include as follows: First, it is consistency with the request format that users are familiar with. Second, if necessary, client program can do future operations easily. Finally, as everyone knows, XML is a standard format for Electronic Data Interchange on the web. It can interoperate with other systems. As we know, the result of some statistical systems is also formatted by XML. Directly viewing XML file is not effortless task. So that the result can also be shown in browser by adding XSLT/XPATH [6] translation into XML file that is done by SA. So that the client can either directly display using browser or further operate the result by other applications without extra effort.

3 Example

In the section we show an example to demonstrate the use of SAs; Example invokes the BSA. The example is excerpted from [15]. Here we only show the XML file and its result of the example. The principle of why use the operations please refers to the [15].

Example
The following data are the ages of a sample of employees from a government department:
42, 53, 61, 20, 28, 48, 47, 42, 38, 39, 33, 27, 36, 40, 44, 44, 35, 35, 64, 52
Compute the mean and variance of the sample data.

The problem belongs to descriptive statistics and needs to invoke the BSA. The XML file is shown in Fig. 6 and the result is shown in Fig. 7

```
<?xml version = "1.0" encoding="Big5"?>
<MASS_SUBMIT>
    <USERID>guest</USERID>
    <AGENTLIST>
        <FROMAGENT>Dispatching</FROMAGENT>
        <TOAGENT>Basic_Statistical</TOAGENT>
    </AGENTLIST>
    <FUNCTLIST>
        <FUNCTION>DS</FUNCTION>
    </FUNCTLIST>
    <OPLIST>
        <OP>
            <OPNAME>Mean</OPNAME>
            <INPARA> ages </INPARA>
        </OP>
        <OP>
            <OPNAME>SampleVariance</OPNAME>
            <INPARA> ages </INPARA>
        </OP>
    </OPLIST>
    <PARALIST>
        <PARA>
            <PARANAME> ages </PARANAME>
            <TYPE>ArrayOfDouble</TYPE>
            <DATA>
                <VALUE>42, 53, 61, 20, 28, 48, 47, 42,
38, 39, 33, 27, 36, 40, 44, 44, 35, 35, 64, 52</VALUE>
            </DATA>
        </PARA>
    </PARALIST>
</MASS_SUBMIT>
```

Fig. 6. XML format of invoking Basic Statistical Agent

Fig. 7. The result of Example.

4 Concluding remarks

In this paper, we propose a flexible and scalable statistical environment based on the web computing and multi-agent technology and offer a unified XML-based programming interface for statistical computing. The system separates various statistical functionalities into groups of services. All of services in the environment are defined as an agent and implemented using web service technology. And all the statistical operations are formulated as a XML document. This is a unified and standardized programming interface and can offer scientists programming their statistical model over the WWW in an easy way. Statisticians can easily program their solution and the programs can be Write-Once and Run-Anywhere (WORA). And the result of MASS is formatted in XML that also is a standard format for Electronic Data Interchange on the web. It can interoperate with other systems to achieve electronic statistics.

In addition, we construct a two level meta-data hierarchy in the MA to specify the design of the core agents. Level I is system level that is statically constructed in configuration time and level II is function level that may be dynamically updated in run time when SAs updating it's service. The metadata offer users and agents the global view of MASS, so that the agent can query the capability and syntax of system, including the format of request and response. New services can be affiliated to the system easily by updating metadata of system. The metadata of system are inquired and maintained in the MA. SA can be added and upgraded without any modification of the system architecture and user interface. Obviously, such functionality can improve system's flexibility and scalability.

Acknowledgements:
This work was supported by the National Science Council of the Republic of China under Grant No. NSC 93-2118-M-305-003 and No. NSC 94-2752-E-009-006 -PAE.. In addition, we would like to thanks Miss. Wu Miao-Yue for his assistance in this project.

Reference:

[1] Chooichiro Asano and Akinobu Takeuchi, "Web-based statistical system by using the DLL," *Computational Statistics & Data Analysis*, 44(2003), 409-418.

[2] Oliver Gonther, Rudolf Moller, Peter Schmidt, Hemant K. Bhargava, Ramayya Krishnan, "MMM: A Web-Based System for Sharing Statistical Computing Modules," *IEEE Internet Computing*, May-June 1997, pp.59-68.

[3] StatCrunch- Statistical software for data analysis on the Web, http://www.webstatsoftware.com.

[4] WebServices – SOAP, http://xml.apache.org/soap/

[5] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, 15 March 2001.

[6] Extensible Markup Language (XML), http://www.w3c.org/XML/.

[7] Net-Stat, http://netstat.stat.tku.edu.tw/

[8] eStat, http://estat.ncku.edu.tw/new/

[9] The R Project for Statistical Computing, http://www.r-project.org/

[10] Junji Nakano et. al., "A statistical package based on Pnuts," *COMPSTAT 2000 Proceedings* (2000), pp. 361-366

[11] Junji Nakano et. al., "An implementation of a statistical language based on Java," *Journal of the Japanese Society of Computational Statistics*, 14 (2001), 59-67.

[12] H.M. Deitel, P.J. Deitel, B. DuWaldt, L.K. Trees, "Web Services: A Technical Introduction," Prentice Hall, Upper Saddle River, New Jersey 07458.

[13]S-PLUS, http://www.insightful.com/support/doc_splus_win.asp

[14]JMSL for Java, http://www.vni.com/products/imsl/jmsl/jmsl.html

[15]Robert V. Hogg, Elliot A. Tanis, " Probability and Statistical Inference," Fifth Edition, PRENTICE-HALL INTERNATIONAL,INC.

[16]SAS/CONNECT software, http://www.sas.com/technologies/dw/etl/connect/index.html.