XML-BASED WebGIS PROTOCOLS*

LUO Yingwei, LIU Xinpeng, WANG Xiaolin and XU Zhuoqun

Dept. of Computer Science and Technology, Peking University, Beijing, P.R.China, 100871

ABSTRACT

Aiming to the features of complex objects and massive data transmission, a new XML-based method to design and implement communication protocols for WebGIS is presented. The paper illustrates the mechanism in designing communication protocols following W3C's XML Schema specification, as well as provides main flow to embed them into WebGIS by packing and parsing XML-based protocols. This kind of protocols can be used in spatial information exchange among heterogeneous platforms of in distributed environment.

KEY WORDS: XML, XML Schema, WebGIS, Protocol

1 INTRODUCTION

WebGIS uses network to distribute geographical information and relevant disposal. It is the outcome of object-oriented software component technology and information interoperation technology, as well as the development of network technology. Nowadays international studies on WebGIS software technology focus mainly on several facets, such as spatial data model, spatial data structure, organization and management of spatial information, communication protocols, distribution strategies of spatial information and services, and so on. The facet discussed in this paper is the subject of the communication protocols for WebGIS [1].

The Communication Protocols for WebGIS generally lies in two kinds: control command protocols and spatial data transmission protocols. Figure 1 illustrates the distributed computing model for WebGIS.

[*] Corresponding author: LUO Yingwei, lyw@pku.edu.cn.

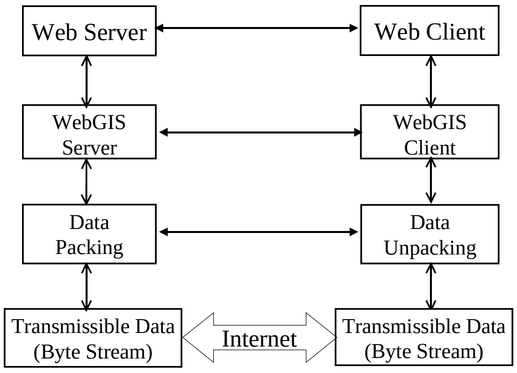


Figure 1 the Distributed Computation Model for WebGIS

The description of control command using parameter frame needs cautious definition for protocol format and different implementation for parsing corresponding protocol, so it does not own the common usage. For example, a parameter frame of a data request protocol may have the definition as following:

RequestUser	RequestTime	RequestWholeLayer	MapName	NeedIndex	IndexID	Other Field…

But in fact, the protocol requesting a layer of "Traffic Lines" always is filled just as following. There are too many null fields.

"lxp"	2002/6/10	True	"Traffic Lines"	False	Null	Null for Other Fields

XML can be used conveniently to describe the concept model of inclusion relationship. Additionally, it can directly express the concept model in an understandable way, and the expression format is so flexible that it seldom suffers the limitation which parameter frame does. While describing protocols by XML, we can not only give a common format for data and control commands, but also reuse the existing XML parsers, so as to facilitate the expansibility and integration of protocols in a system.

At present there already has precedents that used XML to express communication protocols. In the Web server based data release platform - ArcIMS, ArcInfo had already used ArcXML as the fundamental command and data transmission protocols to communicate between users' web pages and backend spatial data servers [2]. More worthily mentioned, W3C had proposed Simple Object Access Protocol (SOAP) 1.1[3] in May, 2000, which is a light weight protocol based on XML used to build information exchange framework under distributed environments. Our idea of XML based communication protocols for WebGIS benefits from the SOAP model, but we basically focus on the application in WebGIS.

2 ANALYSIS OF WebGIS COMMUNICATION PROTOCOLS

2.1 Analysis of Applet Based WebGIS

In order to better analyze the communication protocols for WebGIS, we have developed a WebGIS prototype system that provides simple applications using the platform-independent language – Java. This system consists of three parts: Applet, WebGIS server and data server, as showed by Figure 2.

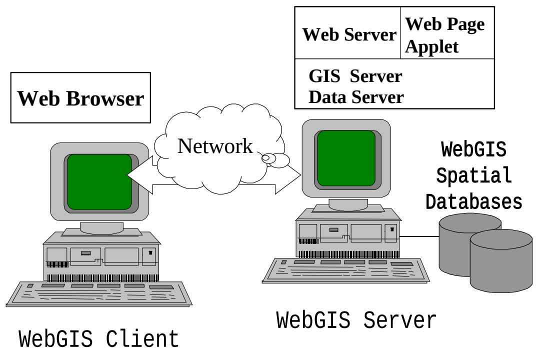


Figure 2 Applet Based WebGIS Architecture

Data server provides the interface for WebGIS to access to spatial databases. WebGIS server is responsible for listening any requests from client and building connections with legal users. GIS server has special database access modules that are responsible for accomplishing interactions with data server and retrieving data information that users request. WebGIS server also integrates several modules related to analysis and disposal of GIS data, such as topological analysis, shortest path analysis, overlay analysis, data conversion and so on. In this way, more complex GIS data disposal tasks will be accomplished at WebGIS server side, while only fundamental map manipulation functions are remained in applets. Such mechanism guarantees a small and exquisite WebGIS client while applets are being downloaded and running, and in the end we can implement “Thin Client”. By embedding applets into web pages and providing many kinds of interaction interfaces together, WebGIS users may fulfill many kinds of spatial manipulation functions.

During the whole interaction process of WebGIS, applets perform as a key role. Through encapsulating fundamental GIS functions into applets, we can build a series of special function-oriented applets as toolkits, such as map visualization applet, topological analysis applet, feature query applet, etc. By means of redefinition to mouse events inspired from these applets’ interfaces, users could refactor these applets using languages such as Java Script, just like we do to the components.

While applet is running, user’s requests and responses from Web server form the basal interactive transactions. These transactions will all be expressed by communication protocols for WebGIS. In order to accomplish elementary GIS interactive functions provided by applets’ interfaces, both client and server have to compartmentalize the details of requests and replies, define a pretty protocol specification, so as to obtain high efficiency and accurate interactions. That is because of:

- ■ In order to utilize network band to a further extent as well as enhance the interactive concurrence of the system, careful analysis on protocol content should be paid for fundamental transactional interactions in WebGIS. Then give strict format by protocol contents, so as to define high quality request and reply patterns.

- ■ The communication protocols for WebGIS have an obvious structure of hierarchy, so traditional protocol format (parameter frame) may result in a rather complex protocol set carried with too many redundant linear fields, and leads to a complex protocol development and a decline of communication efficiency. So defining a new protocol specification becomes an urgent problem.

- ■ The communication protocols for WebGIS should have flexibility to be easily updated or changed. Also, the new protocol specification should take account dynamic scalability of the protocols.

- ■ After analyzing fundamental interactions in WebGIS, we have discovered that the communication protocols for WebGIS possess a typical tree-structure, thus they can be conveniently modeled by class hierarchy and normalized by formal UML diagrams. Class inheritance and composition mechanism provide favorable structural support to expand the protocols, so it is possible for us to define a bran-new protocol specification.

Interactions between users and WebGIS client determine main contents of the communication protocols for WebGIS. The following two sections show detailed illustrations of client request and server reply protocols as well as formal descriptions by UML diagram.

2.2 Illustrations of WebGIS Request Protocols in UML

In the prototype system, modules such as map manipulation and entity query are contained inside applets. So the communication protocols for WebGIS is mainly responsible for requests and replies of map data. A map consists of several layers, and the basic request is to ask for a layer. There are different ways to request a layer, which mainly include:

- ■ Requesting a layer by providing the layer’s name, this is the most basic way;

■ Requesting a layer by specifying the layer’s redirection address;

■ Requesting a block of geometry entities in a layer by providing a spatial index sub tree;

■ Requesting a block of geometry entities in a layer by specifying a spatial range.

According to above analysis on primary contents of layer request, figure 3 presents the UML descriptions of map data request protocols for WebGIS.

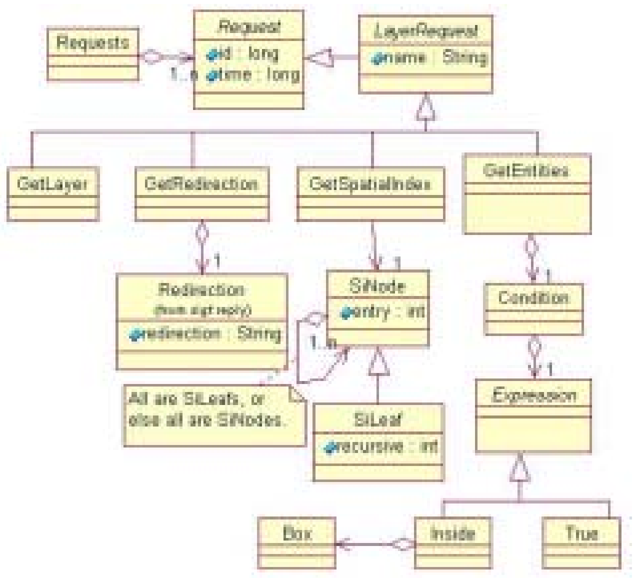


Figure 3 UML Descriptions of Request Protocols

Each request starts from root class ***Requests***, which acts as a container of series of ***Request***. Class ***Request*** should have an *id*, as well as a *time* recording the request time. Class ***LayerRequest*** derived from ***Request*** denotes that the current protocols we designed only contain layer request. Every ***LayerRequest*** should contain a necessary attribute *name*, which indicates the layer name. As described above, there are four classes corresponding to four request ways - ***GetLayer***, ***GetRedirection***, ***GetSpatialIndex*** and ***GetEntities***, all derived from ***LayerRequest***.

■ ***GetLayer*** will retrieve whole layer data identified by the layer name;

■ ***GetRedirection*** needs a member class ***Redirection*** to provide the redirected data address *redirection*;

■ ***GetSpatialIndex*** needs a spatial index sub tree, which is used to express index description of a block of geometry entities that users request in a layer. The spatial index sub tree is built up by basic tree node ***SiNode***, which could be just a leaf node ***SiLeaf*** with the attribute *recursive* identifying its recursive tiers, or may be another sub tree with the attribute *entry* identifying the entry for each non-leaf node in it.

■ ***GetEntities*** needs member class ***Condition*** to describe users’ restriction for entities in a layer. The restriction is expressed by <u>***Expression***</u>, which may either be <u>***True***</u> which denotes no restriction, or be an <u>***Inside***</u> specifying a spatial range by <u>***Box***</u>.

2.3 Illustrations of WebGIS Reply Protocols in UML

Aiming at above specification of WebGIS request protocols, replies from server may have a corresponding design, which mainly include:

- ■ For <u>***GetLayer***</u> request, which is determined only by the layer name, the reply is whole layer data of the layer.
- ■ For <u>***GetSpatialIndex***</u> request, the reply is corresponding spatial index sub tree with each node carrying with index information;
- ■ For <u>***GetEntities***</u> request, the reply is all information of geometry entities located in the specified spatial range or a redirected address of some fields of those entities;
- ■ For <u>***GetRedirection***</u> request, the reply is simply a redirected address, and redirected data will be retrieved by corresponding disposal modules of server;
- ■ For the request of getting a single field of geometry entities, the reply is a redirected address of the field of those entities.

Figure 4 illustrates the class hierarchy of the reply protocols in UML diagram.

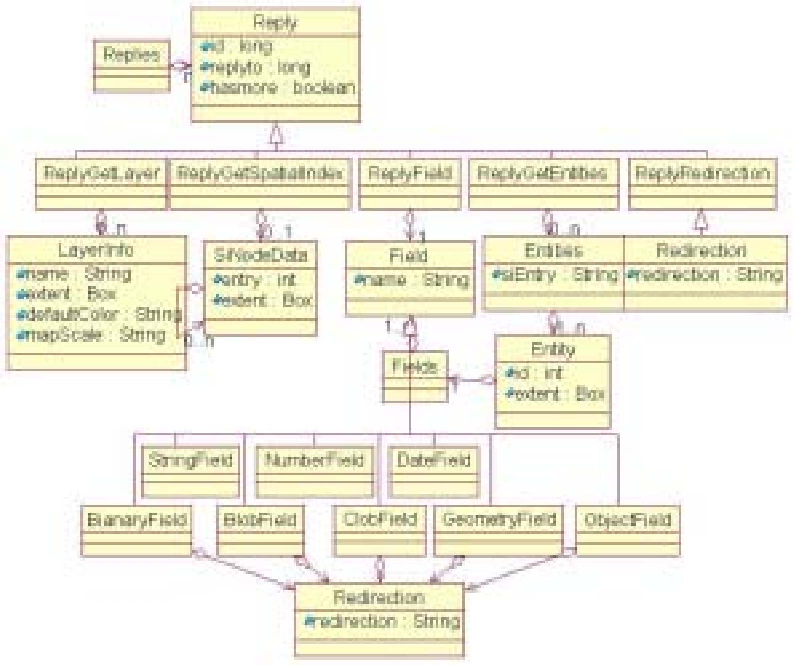


Figure 4 UML Descriptions of Reply Protocols

Just like the request protocols, class <u>***Replies***</u> is a container of <u>***Reply***</u>, which can represent different replies from server. Class ***Reply*** must have an *id* to identify itself, meanwhile it must specify the reply-to object by *replyto*, and use *hasmore* to denote whether there are still other replies or the current reply is the last one. There are five concrete classes of replies inherited from class ***Reply***: ***ReplyGetLayer***, ***ReplyGetSpatialIndex***, ***ReplyGetEntities***, ***ReplyRedirection*** and ***ReplyField***.

- ***ReplyGetLayer*** is a reply to ***GetLayer***, which includes basic information of required layer: layer name, layer extent, default entity Color, map scale and entities;
- ***ReplyGetSpatialIndex*** is a reply to ***GetSpatialIndex***, which contains a spatial index sub trees with tree nodes (leaf or non-leaf) of type ***SiNodeData***. Each node lists its *entry* and the range of its sub tree (*extent*);
- ***ReplyGetEntities*** is a reply to ***GetEntities***, which is an ***Entities*** containing a list of ***Entity*** that satisfy request restriction. Each ***Entity*** has an *id* to identify itself and marks its range by *extent*. A ***Fields*** depicts detail information of an ***Entity***. ***Fields*** records a list of redirection information for different ***Field*** of ***Entity***. ***Field*** can be embodied as eight classes: ***BianaryField***, ***BlobField***, ***ClobField***, ***GeometryField***, ***ObjectField***, ***StringField***, ***NumberField*** and ***DateField***.
- ***ReplyRedirection*** is a reply to ***GetRedirection***, which contains a ***Redirection*** specifying redirected address by string *redirection*;
- In addition, ***ReplyField*** could be used to directly provide the reply of the request of getting a certain field of geometry entities.

3 IMPLEMEMENTING XML-BASED WebGIS PROTOCOLS

When proposing definitions of XML Schema for the protocols, we also define metadata for the protocols. Using metadata to normalize the protocols may define basic expressions of the protocols at pattern-abstract level of data. The rewrite of metadata can accomplish the redefinition of a set of protocols, which is useful to the update and expansion of a certain set of protocol.

3.1 Defining XML Schema (Metadata) for the Protocols

We can conveniently convert class hierarchies in UML to corresponding expressions in XML Schema. The conversion is usually based on the following rules [4][5][6](for the limitation of length, detailed XML Schema of request and reply protocols can be obtained at: http://gis.pku.edu.cn/Projects/WebGIS/protocol/):

- A class in UML has a counterpoint - a complex type in XML Schema. The classes in UML may be divided into two kinds: abstract class and non-abstract class. An abstract class usually generalizes common attributes and methods of those classes inherited from it, but itself cannot have an instance. In XML Schema, the ***abstract*** attribute is used to identify corresponding complex type of abstract class;

■ Attributes of class in UML is equal to those of corresponding complex types in XML Schema;

■ Class inheritance in UML is denoted by the value of ***base*** attribute, which is an extended mark in XML Schema. The value specifies the type of base class in an inheritance chain;

■ Member class in UML is expressed by nesting sub-element, which is a complex type in XML Schema.

3.2 Validator for the Protocols

3.2.1 Application of Validator in Transmission of WebGIS Protocols

One of the main aims of using XML Schema to describe communication protocols for WebGIS lies that, utilizing the metadata describing capacity of XML Schema to XML documents, a common XML Schema based XML document validator can be used to automatically perform validity checking of XML based communication protocols. The validator can be regarded as a process switch at both client and server side during the transmission of the protocols. In order to enhance system efficiency, we adopt following premises:

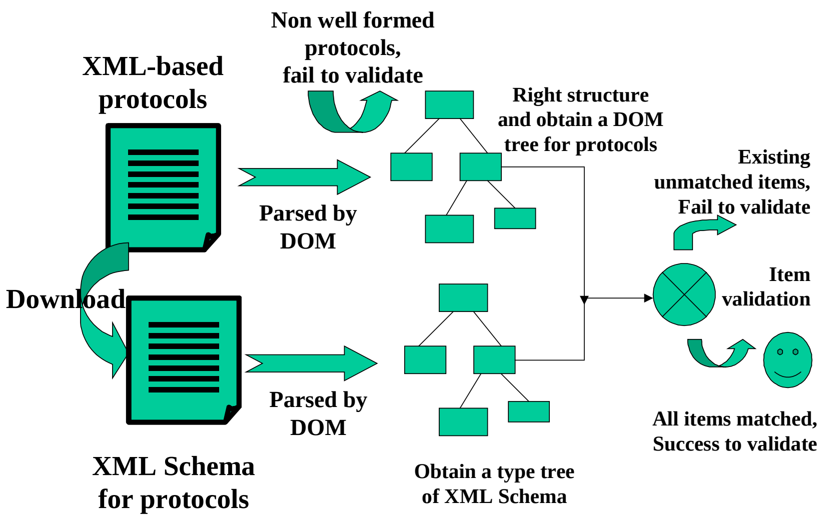
■ Reply protocols from server side are usually considered to strictly follow XML Schema definition above, so when client side receive the reply, it is allowed to close validation switch and directly perform parsing XML based protocols;

■ Familiar and friendly clients will be recorded at server side. The protocols sent from there are also usually considered to strictly follow XML Schema definition, so when server deals with these requests, it is allowed to close validation switch and directly perform parsing task;

■ A new client must experience a certain phase to win the confidence of server. During this phase, server must check request protocols sent from the new clients, so as to avoid unnecessary waste of parsing resources that is brought by protocols with mistakes.

3.2.2 Work Flow of Validator

Work flow of validator is expressed in Figure 5. Our system used DOM (Document Object Model) to parse XML Schema of the protocols and themselves. Although using XML Schema to check validity of the protocols guarantees parsing them later correctly, but the work flow is very complex and time-consuming, so validity checking is executed only when it is highly required.



XML-based protocols
Non well formed protocols, fail to validate
Right structure and obtain a DOM tree for protocols
Existing unmatched items, Fail to validate
Parsed by DOM
Download
Item validation
Parsed by DOM
All items matched, Success to validate
XML Schema for protocols
Obtain a type tree of XML Schema

Figure 5 Work Flow of XML Schema-based Validator for Protocols

3.2.3 Elementary Validation Items in XML Schema of the Protocols

■ Check for nesting structures of an element type

The primary method to implement checking for nesting structures of an element type is: starting from root of the DOM tree of a protocol and root of the type tree of its XML Schema, and using depth-first search algorithm to check each node of DOM tree using the node information of corresponding type tree. This work is the framework of following validation items. During traversal of the trees, each node should have following items checked.

■ Check for element name

All element names occurring in the DOM tree should have corresponding type definitions in the type tree of XML Schema. Additionally, because of the match case feature, the name of each element in the DOM tree must match case with corresponding definition in the type tree.

■ Check for attribute inheritance and the ***use*** attribute

The construction of the type tree facilitates checking for attribute inheritance. What validator should do is to match attribute list of element node in the DOM tree with attribute list of corresponding element definition in the type tree. While matching, we should verify whether required attributes presented in expression of the protocols, whether attribute names and attribute types matched. Required attributes are denoted by value of the ***use*** attribute in XML Schema, which are also recorded in the type tree.

■ Check for occurring times (the ***minOccurs***, ***maxOccurs*** attributes) of an element

Validator maintains a counter for each element in the DOM tree, and during traversal of the DOM tree, the counter will record actual appearance times for an elements. At last, the appearance time is compared with the corresponding number defined in the type tree. If mismatch, a protocol will be considered invalid.

- Check for content arrangements (the ***sequence***, ***choice***, ***all*** attributes) of a complex element

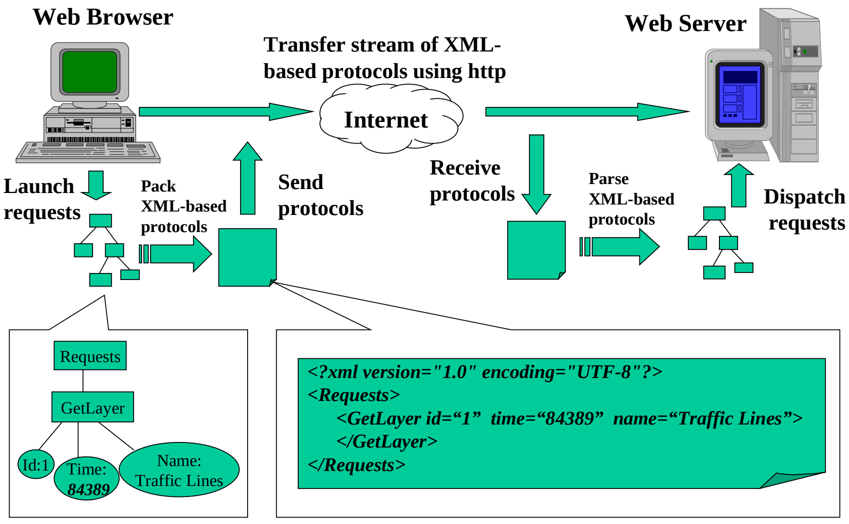
Content arrangements supported by XML Schema include ***sequence***, ***choice*** and ***all***. In the type tree of XML Schema, ***sequence***, ***choice*** and ***all*** will present as an independent node. Validator will produce an element chain for their children of these three nodes, and use the appearance order and appearance times of corresponding elements to perform checking.

3.3 Packing into XML and Parsing to Objects for the Protocols

By defining XML Schema for the protocols, we can present their metadata. By implementing XML Schema based validator, we can guarantee the correctness of the protocols before transferring them by HTTP. But if we wish to embed XML-based protocols into WebGIS, we should have concrete request or reply protocols packed from object set into an XML stream as well as parsed from XML stream back into object set at both client and server side.

3.3.1 A Simple Request Conversation

Figure 6 describes a sample for conversation flow of packing and parsing the protocols. A user launches a request of get layer “Traffic Lines of Beijing”. Firstly, an object tree for the protocol (***GetLayer***) should be constructed the in the memory using UML class descriptions. After packing the object tree into XML, we get an XML character stream that satisfies XML Schema specification of ***GetLayer***. The stream arrives to server side through Internet connection between C/S, and then is reconstruct to an object tree. After that, server recognizes the request is a ***GetLayer*** request, and dispatches to GetLayer-processing module. GetLayer-processing module queries database by attributes of ***GetLayer*** to retrieve the layer “Traffic Lines of Beijing”.



```
<?xml version="1.0" encoding="UTF-8"?>
<Requests>
    <GetLayer id="1"  time="84389"  name="Traffic Lines">
    </GetLayer>
</Requests>
```

Figure 6 Packing and parsing of XML-based Protocols for WebGIS

3.3.2 Packing into XML

Packing the protocols into XML exists in protocol-processing modules at both client and server side. At client side, request events from applets triggered by users are captured and stored as object trees. Object trees of requests are instance set of Java classes designed by UML diagrams for the protocols. It acts as middle interface between user requests and XML construction of the protocols, and is an important part in embedding XML-based protocols into WebGIS. At server side, result data of each processing module and reply information also need to be stored as object trees, and then are packed into an XML character stream and sent back to client side.

The packing process is simple, which is similar to the conversion of UML diagrams into XML Schema of the protocols. The difference is that the latter accomplishes the mapping of objects to XML syntax at metadata level, while the former does a same thing at data level.

3.3.3 Parsing to Objects

A complete conversation from client to server and then back to client must include the converse process of packing the protocols into XML - parsing the protocols to objects. It also exists in protocol-processing modules at both client and server side. From the viewpoint of implementation, parsing the protocols to objects is similar to validating the protocols. Firstly, it is needed to interpret the received XML stream of a protocol to a DOM tree by XML parser, and then retrieve parameters of the protocol from the DOM tree, assemble them back into an object tree of the protocol.

As described above, packing and parsing the protocols embody the integration of XML-based protocols in WebGIS. These guarantee the scalability of the protocols and the integrality and validity of conversation between client and server at implementation level.

4 SCALABILITY AND LIMITATION ANALYSIS

4.1 Structural Characteristics of the Protocols

From Figure 3 and Figure 4 we discover that, the structure of communication protocols for WebGIS possesses favorable scalability because of the usage of class inheritance and composition:

- The encapsulation of attributes and methods of object-oriented mechanism guarantees stability of protocol modules. In fact, an update to internal attributes of one class does not affect any other classes, while previous linear-field based protocol frame does not possess such advantage: an addition or deletion of one field will lead to a series of offset changes to other fields and redesign of the disposal process;

- Class inheritance gives a high level generalization of a set of similar requests or replies. Class inheritance distills common attributes to make protocol expression concise. During early phase of implementation, the protocol set may receive many expansions with the extension of applications and the refinement of the functions. The action to add a new inherited class is independent to any other brother classes, which keeps the validity of the existing protocols to an extreme extent. For example, to add a new class ***mapRequest*** inherited from abstract class ***Requests*** only needs to define ***mapRequest*** itself.

- Class composition expresses concrete content of a request or a reply, where alteration is taking place frequently. Using fundamental class structures is a very flexible way. When we wish to reference or change the content, the only thing we should do is to alternate the member classes of a request or a reply. Here is a convictive example: very complex communication protocols for WebGIS may include request like ***MultiLayerRequest***, which may request several layers in a map at one time. ***MultiLayerRequest*** will be used to do overlay analysis on relevant layers at client side. ***MultiLayerRequest*** inherits from ***Requests***, so it should have a ***LayerRequest*** as its member class, although it may contain some other attributes to identify relationship among requested layers. Here, ***MultiLayerRequest*** can achieve reference to whole structure of ***LayerRequest*** only using ***LayerRequest*** as its member class.

4.2 Application Limitations of the Protocols

The limitations of XML-based communication protocols for WebGIS result from the structural disposal of XML character streams.

Comparatively powerful parser for XML is DOM, which can interpret an XML document to an object tree. We can conveniently use methods of DOM to retrieve element names, element attributes, and text contents from XML document.

We use Java package for DOM in our development. While due to the huge size of the package and the relative low version Java cores embedded in the Microsoft IE, WebGIS client is forced to download the package so as to correctly perform protocol parsing and packing. This weakens the tendency to implement a thin client. So application of DOM to client is confined. Presently the feasible solution is as following:

- ■ In client, we can use another XML parser – SAX (Simple API for XML) to perform parsing task. SAX is rather smaller than DOM, and provides the power to identify beginning mark, ending mark, text mark and other parts of each element in an XML document. It also provides event redefinition interfaces to implement element parsing. Compared with DOM, the power of SAX is much weaker: it couldn’t identify whole nesting structures of elements in an XML document, but only defines common event interfaces for all elements. For developers, they should deal with events themselves to let SAX reach a similar function of DOM;

- ■ We can also directly convert XML character streams at client side. That is to say, taking each protocol object as a unit, we emit corresponding XML string of the object into output stream of connection according to XML specification and the XML Schema formats for the protocols. Because classes related to the current protocols almost have a mapping to XML elements and its number is not so large, this way is feasible. But with the expansion of the protocols, the complexity will sharply raise if we can’t find a suitable alternate tool for DOM.

5 CONCLUSIONS

Combining with application backgrounds of WebGIS, this paper proposed a basic thought of expressing communication protocols for WebGIS using XML. The primary work we have done in this paper includes:

- ■ Presenting fundamental framework and UML descriptions of requests and replies protocols for WebGIS;

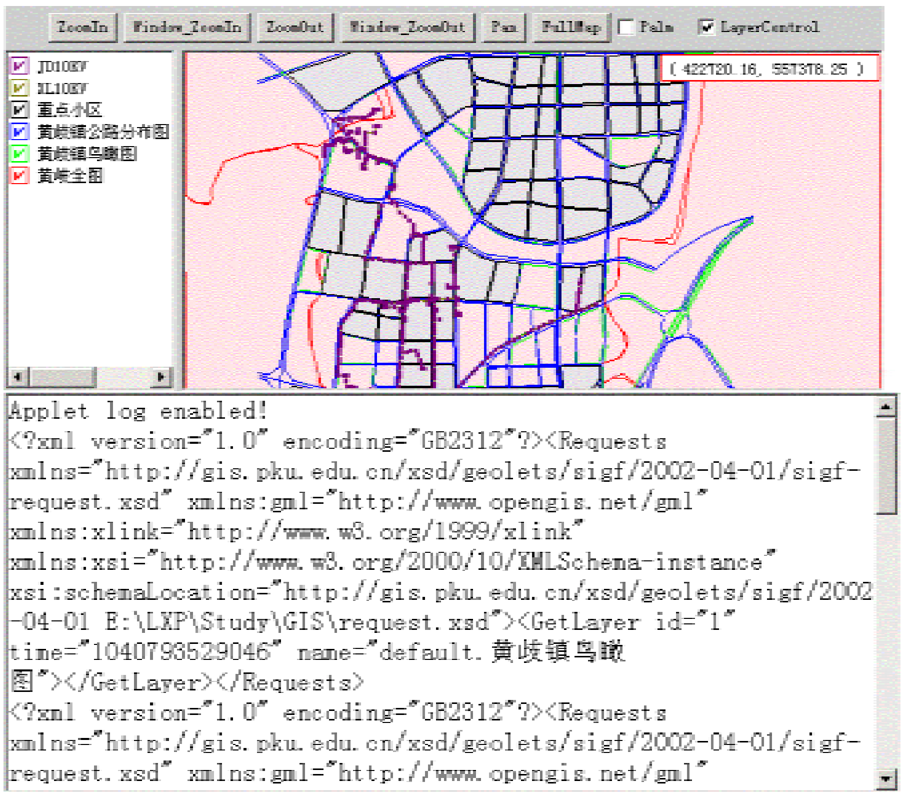
- ■ Based on XML and XML Schema, Implementing transform of protocols from UML description to their definition of XML Schema;

- ■ Giving a concept of implementing protocol-relevant validator;

- ■ Indicating roles of XML packing and parsing during the whole process of protocols for WebGIS by an instance;

- ■ Analyzing advantages and limitations of XML-based communication protocols for WebGIS.

We have already implemented basic map manipulation functions at client side in prototype system. Figure 7 is a screen shot of the system at client side. At the same time of map visualization, XML-based communication protocols for WebGIS are printed out in below text field.



```
Applet log enabled!
<?xml version="1.0" encoding="GB2312"?><Requests
xmlns="http://gis.pku.edu.cn/xsd/geolets/sigf/2002-04-01/sigf-
request.xsd" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://gis.pku.edu.cn/xsd/geolets/sigf/2002
-04-01 E:\LXP\Study\GIS\request.xsd"><GetLayer id="1"
time="1040793529046" name="default.黄岐镇鸟瞰
图"></GetLayer></Requests>
<?xml version="1.0" encoding="GB2312"?><Requests
xmlns="http://gis.pku.edu.cn/xsd/geolets/sigf/2002-04-01/sigf-
request.xsd" xmlns:gml="http://www.opengis.net/gml"
```

Figure 7 A Demonstrative Screenshot of Prototype System at client side

XML-based communication protocols for WebGIS are proposed as a new design concept and have been given an elementary implementation, but up to date the protocols could only perform simple C/S interactions, and the further work includes:

- Studying the details of communication conversation for WebGIS based on the protocols, and implementing multi-interactions during one transaction;

- Implementing concurrent transmission, correct interception and dispatch of XML protocol streams at both client and server side, so as to improve communication efficiency of the system;

- Taking account of the requirements of further development at WebGIS client side using applets, to expand and define highly flexible protocol sets as well as to accomplish definition of XML Schema metadata;

- Refining protocol validator mechanism using common XML Schema based validator for XML.

ACKNOWLEDGEMENTS

Supported by the National Natural Science Foundation of China under Grant Nos. 60030005, 60073016, 60203002; the National High Technology Development 863 Program of China under Grant Nos. 2001AA113151, 2002AA135330, 2002AA134030; the National Grand Fundamental Research 973 Program of China under Grant No. 2002CB312000; the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20020001015; the Natural Science Foundation of Beijing of China under Grant No. 4012007.

REFERENCES

[1] Li Muhua (2000). "Study and Implementation of WebGIS as a Component System" [Master Dissertation] (in Chinese). Beijing: Peking University, 2000.5

[2] ESRI (2001). ArcXML Programmer's Reference Guide (ArcIMS 3), PDF on CD only, 2001.4.

[3] W3C Note (2000). "Simple Object Access Protocol (SOAP) 1.1", http://www.w3.org/TR/SOAP, 08 May 2000.

[4] W3C Proposed Recommendation (2001). "XML Schema Part 0: Primer", http://www.w3.org/TR/2001/PR-xmlschema-0-20010330, 30 March 2001.

[5] W3C Proposed Recommendation (2001). "XML Schema Part 1: Structures", http://www.w3.org/TR/2001/PR-xmlschema-1-20010330, 30 March 2001.

[6] W3C Proposed Recommendation (2001). "XML Schema Part 2: Data types", http://www.w3.org/TR/2001/PR-xmlschema-2-20010330, 30 March 2001.