GEO-METADATA CREATION USING GML TECHNOLOGY

A. Kalantari Oskou'ee, M.S.
East Azarbaijan Research Center for Natural Resources
email: oskhom@yahoo.com

Abstract – Nowadays, standardized metadata for geospatial data is a key in sharing and finding information on the web and crucial in building Geospatial Data Infrastructure (GDI). The main objective of this paper was to develop a web-based metadata dissemination system for in-situ sensors based on, most importantly, interoperable, standard and open technologies introduced by Open GIS Consortium (OGC), namely Geography Markup Language (GML). In this research, at first a use case diagram was developed to demonstrate the user's requirement. Then, an application XML schema based on user's requirement was created. To build this schema, some GML schema documents (developed by Open GIS Consortium (OGC)) were imported into the application schema. System architecture was designed based on client/server model and a UML class diagram was also developed to present all classes and their attributes, operations and associations within the system. Implementation was conducted using GML, XML, XMLHTTP, DOM, ASP, and VBScript that brought out a web-network-based in-situ sensors metadata application. This application provided a user friendly interface to search and find sensor related information. Results showed that although GML and XML are powerful tools to build geo-metadata, it is important to note that GML document size may be a problem when dealing with huge amount of data.

Keywords: Metadata, GML, In-Situ Sensor, Interoperability, OGC.

INTRODUCTION

Metadata is defined as "data about data". It is the background information, which describes the content, quality, condition, and other appropriate characteristics of the data. The term is generally applied to electronic resources. Metadata can be categorized into several levels ranging from a simple list of basic information about available data to detailed documentation about an individual data set [3]. In order to build a strong geospatial data infrastructure (GDI), metadata is crucial. Metadata and its servers enable users to integrate data from multiple sources, organizations, and formats. Metadata for geographical information may include the source, date of creation, format, projection, scale, resolution, and accuracy. Metadata can be used internally by the data producer to monitor the status of data sets, and externally to advertise potential users through a national clearinghouse [2].

The main aim in this research is to develop a web-based sensor metadata dissemination

system prototype based on interoperable technologies namely Geography Markup Language (GML). GML is an XML-based data format and one of the most important elements in interoperability program developed by Open GIS Consortium (OGC) [7]. It provides standardized components for the modelling, transport, and storage of geographic information [6].

The motivation for attention and developing metadata arises from produced difficulty in finding information on the web. Now, rapid growth in the amount of web-based resources demands for a powerful mechanism and provides facilities to discover the available web resources in an easy and standard way. Although there are several search engines on the web, searching through sensor metadata dissemination prototype will provide more benefits to user than searching an engine site (such as Yahoo, Google, and so on) in which user must open one by one suggested pages and examine whether there is any desired information or not. It is obvious that this practice is tedious and time-consuming.

Generally, two types of metadata attributes can be identified: human readable metadata in order to be used by the user, and metadata for automation (machine-understandable) [2]. Here, the focus is on the first aspect of metadata namely human readable metadata that provides capabilities for the users to find different data sensors based on their metadata.

Based on literature review, it was realized that there is no study or the work regarding sensor metadata creation using interoperable technologies. The existing systems are based on proprietary technologies so it can be said that this research is the first study in this regard.

MATERIAL AND METHODS

In general, the research methodology is classified into the following steps:

- Analysis and design of the sensor metadata dissemination system
- Implementation

Each of these is described in the following.

SYSTEM DESIGN CRITERIA

The sensor metadata dissemination system was designed with maximum reliance on existing technologies and standards. To develop this system, the following criteria were considered:

- Interoperable data format
- Open data format
- Platform independent data format
- Loosely coupled data format
- Vendor neutral data format

- Flexible and extensible data format
- Use of standardized and interoperable protocol to communication between client and server applications
- Use of client-server Architecture
- Use of XML program using the Microsoft XML parser.

ANALYSIS AND DESIGN

- UML USE CASE DIAGRAM DESIGN

Developing sensor metadata prototype knowledge about the users and user requirements plays an important role for constructing a powerful system. The following UML use case diagram is developed to show the users and describes what users intend to use the system for.

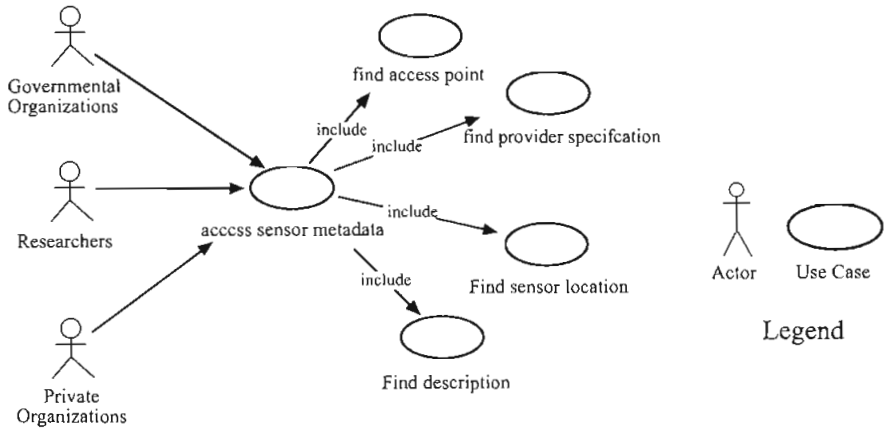

Figure 1: UML Use Case diagram for sensor metadata prototype.

As the use case diagram shows, researchers, governmental and private organizations constitute the sensor metadata system users. Governmental organizations are the major users of sensor metadata system. In all countries over the world, governmental organizations play an important role in managing and controlling for instance, natural resources, air quality, traffic, security, etc. which are based on data measured by different kinds of sensors deployed in environments like rivers, cars, boats, forest, streets and roads. On the other hand, companies can take benefit of sensor metadata system in order to locate the sensor datasets of interest for using in their projects. In general, researchers and research and educational institutes, Location Based Services (LBSs), Protection environment agencies, Meteorology organizations, Municipalities, Police centers, Agricultural organizations, and many other institutes can make use of the sensor metadata system.

DESIGN OF APPLICATION XML SCHEMA

The stage of designing XML-based systems development begins by developing the

applicable XML schema, which defines elements of XML document used by the application. Schema as design tools, establish a framework on which implementation can occur [9]. As mentioned earlier, the sensor metadata prototype is to be based on XML technology to encode and transfer data on the web. To this end, design phase includes first developing an XML schema considering user requirements, which defines XML document elements and then using this schema to edit and validate XML document of prototype. XML schema is a kind of document that defines content model of an XML document. Content model describes which elements and attributes can appear in an XML document, in what order, and how many times [9]. In fact, an XML schema can be thought of as metadata (essentially, data that describes data) for an XML document. To build applicable XML schema for sensor metadata, several GML schema documents are imported into metadata application schema. Here, Altova XMLSPY software is used to schema creation and editing. Based on the system requirements analysis, the following XML schema is developed to support the sensor metadata dissemination system.

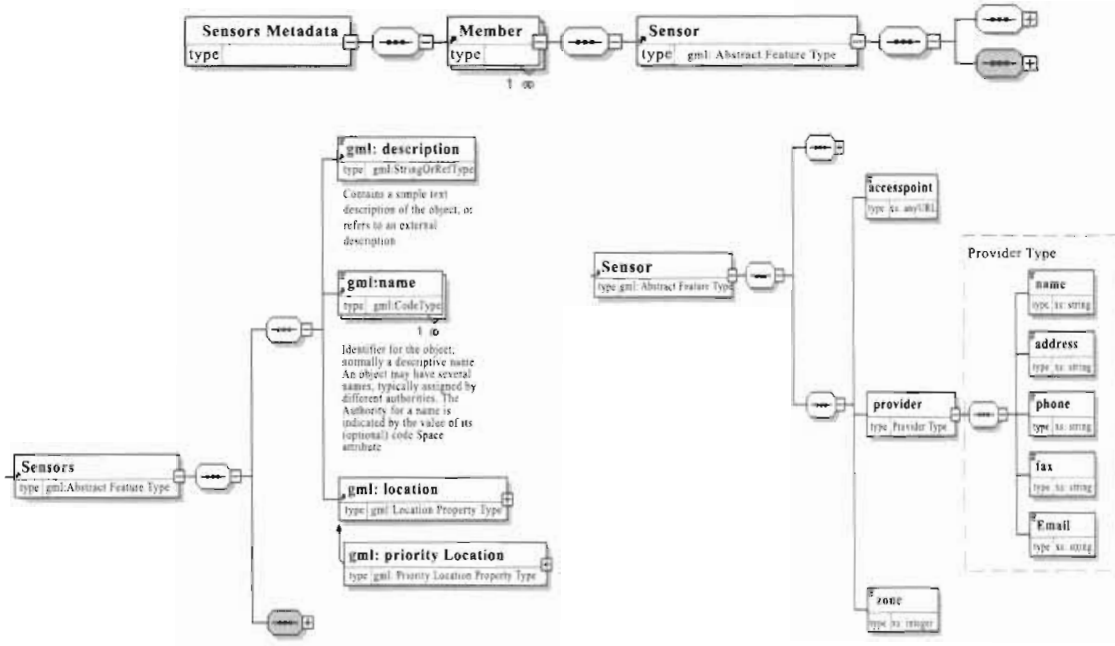

Figure 2: Components of sensor metadata XML schema.

DESIGN OF UML CLASS DIAGRAM

A UML class diagram is used to represent all the classes and their attributes, operations and associations within a sensor metadata system. It models the static aspects of this system [8]. To develop a class diagram, the first step is to identify classes in the sensor metadata system. Since sensor metadata prototype corresponds with server/client model, there are two classes as the following UML class diagram presents:

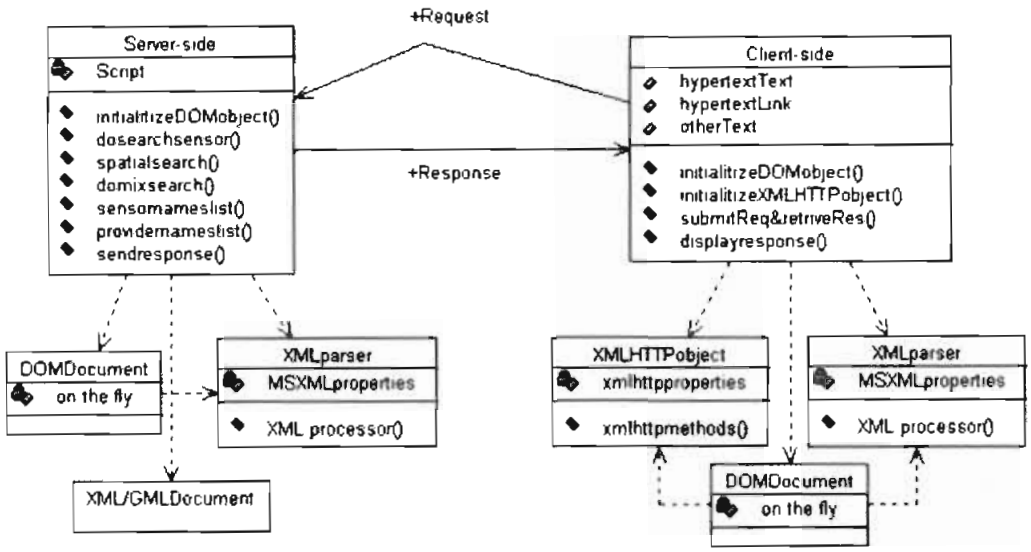

Figure 3: UML class diagram of Sensor Metadata prototype.

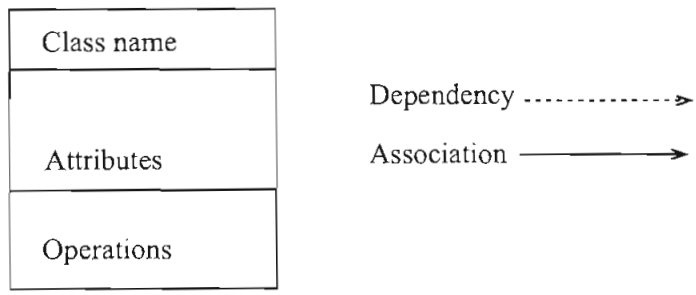

Figure 4: Legend of class diagram.

Client-side class defines attributes and operations that will be used in the client-side browser. Figure 3 also shows relationship between this class and DOM, XMLHTTP objects and XML parser classes. Based on this model, client-side class makes request and submits it to server-side for further process. This relation is shown through association line between two classes in the diagram. The main operation in the client side includes operation in order to initiate DOM and XMLHTTP objects, submit request, retrieve and display responses sent by server-side.

Server-side class defines functions required to support client-side requests. As the class diagram shows, server-side class is responsible to provide the response of request for client-side. Server-side, DOM object, XML parser and XML document are interdependent. The main functions of the server-side are:

- Initializing DOM object, which enables system applications access and manipulates XML/GML document.
- Doing search sensor, which is used to do search against XML/GML document in order to find sensors metadata based on the sensor name user input.
- Spatial search, which is used to do search against XML/GML document in order to

find sensors metadata located in an area.

- Doing mix search, which is used to do search against XML/GML document in order to find sensors metadata based on combination of parameters.
- Sensor name list, which is used to provide a list of registered sensor names available in the system repository.
- Providing name list, which is used to provide a list of registered data provider names available in the system repository.
- Sending response, which is responsible to send the query results toward client-side as XML packets.

UML class diagram was developed using Rational Rose software.

SYSTEM ARCHITECTURE DESIGN

Figure 5 shows developed architecture for sensor metadata dissemination system. This architecture is based on client-server model. According to client-server model, a client using a web browser, interacts with the server-side application, makes a request and submits the request to server-side for further processing. Then, the server-side application retrieves and processes the request and finally sends the relevant response to client [5]. An important point regarding this architecture is that it makes use of XML/GML technologies.

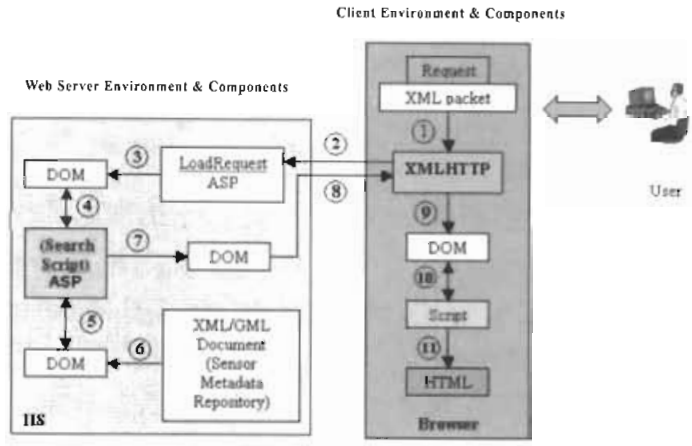

Figure 5: System architecture.

DESIGN OF DATA PROVIDERS ENTRY INTERFACE

Sensor metadata dissemination system is a mediator, which links the sensor data providers and sensor data users (consumers). Here, the aim is to develop a special interface allowing data providers to register their datasets metadata. In fact, the aim of developing this interface is to support the repository of sensor metadata dissemination system via the Internet. The following figure shows the design and mechanism required for this end.

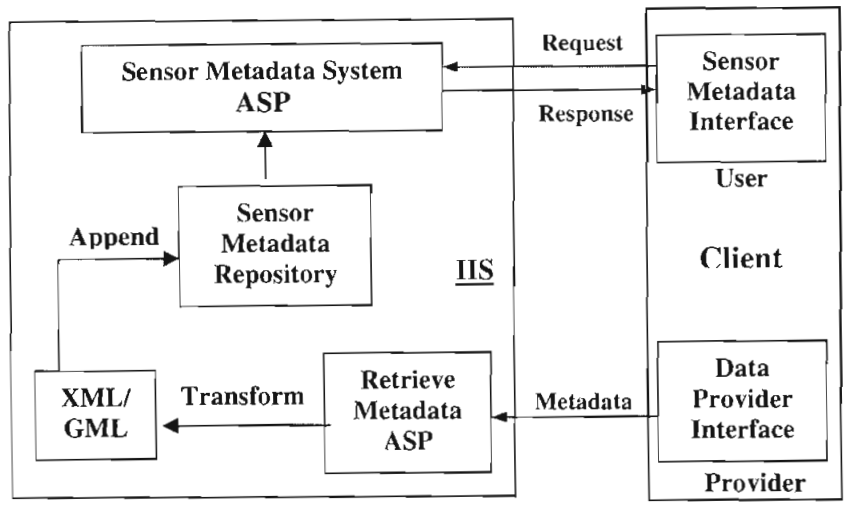

Figure 6: Relationship between sensor metadata prototype, user, and data provider.

Based on this design, data provider starts the scenario and using the provider interface, registers the information of datasets metadata in server side XML/GML-based repository. The data provider's interface can be reached using an URL. Next, the server side script retrieves the received information and using the relevant function converts the received information into an XML/GML format (This process is done since sensor metadata repository has been built based on XML and GML data structures). After completing the transformation step, received metadata is added into sensor metadata prototype repository. This repository is used by the sensor metadata prototype to search based on the request received from the users.

USER'S INTERFACE DESIGN

Similar to other dissemination systems, there is a need for a user interface to provide interaction facilities for the users of the system. The principles of user interfaces design are that user interfaces should be clear, concise, easy to operate, and easy to navigate [1]. Here, the interfaces of the system are designed in a way that every part of the interfaces has a title and a small text describing which main functionality it will supply to fulfil tasks, and each user interface has a return button so that the user can easily go back to the main interface.

This interface provides three clickable options for the users to do search. Available options include: search By Sensor Name, Spatial Search (by window) and search By Name of Sensor, Data Provider and a free Keyword (Advanced Search).

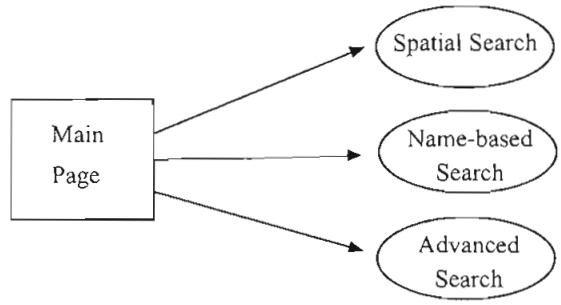

Figure 7: User's interface structure.

IMPLEMENTATION

XML/GML DOCUMENT CREATION

One of the steps in the implementation is to create, edit, and validate XML document. As mentioned previously, XML schema, built in the design section, is used to develop the sensor metadata system XML/GML document. This document is responsible for storing members of sensors metadata. Validation means the process by which an XML instance document is systematically checked and verified to be in conformance with all the rules and restrictions defined within a content model [4]. The following figure shows partially the built XML document for sensor metadata prototype. This document contains root element, three Member elements (the first one is expanded) and Sensor sub element children shown in the grid view of XMLSPY.

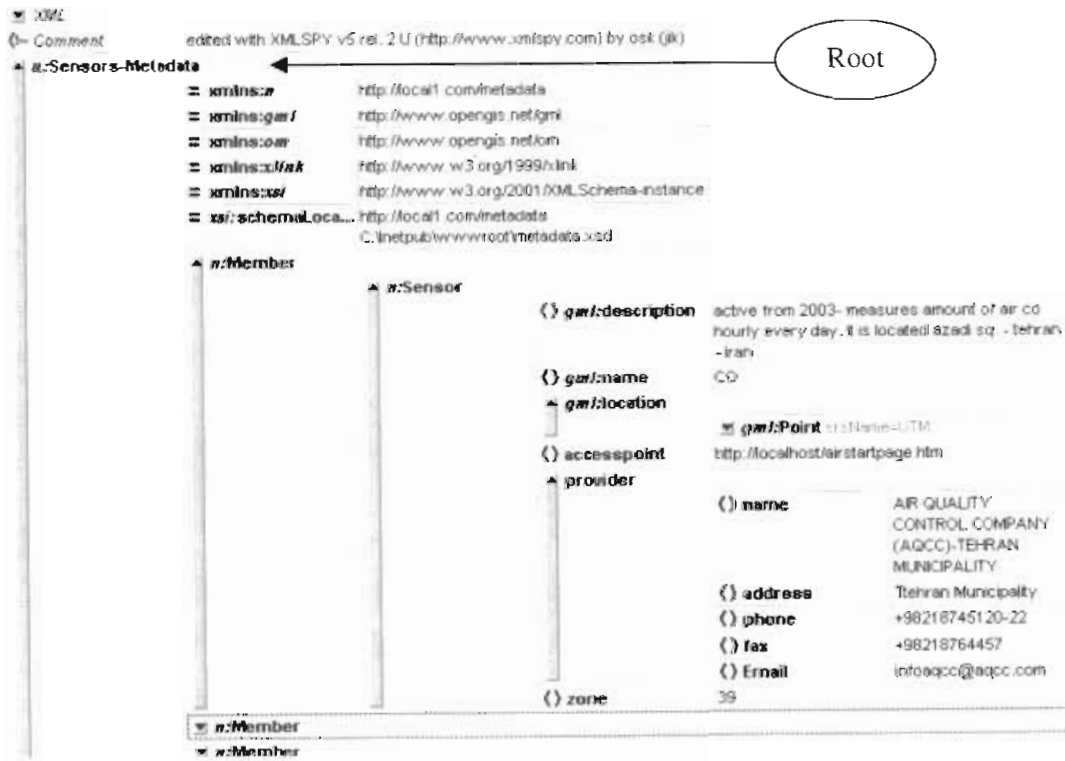

Figure 8: Part of developed XML document for sensor metadata prototype in XMLSPY grid view.

SYSTEM ARCHITECTURE IMPLEMENTATION

Implementation in this phase is classified into two parts: Client side Implementation and server side Implementation. Each of these is described in the following:

CLIENT SIDE IMPLEMENTATION

Based on system architecture, sensor metadata system makes use of XML for sending and receiving messages. In order to pass user input to server side, user input should be packet as an XML packet [1]. The following code is an example used in developing system.

```
xml="<sizeofwindow>"
xml = xml& "<x1value>"& document.rsqfrm.x1.value&"</x1value>"
xml = xml& "<y1value>"& document.rsqfrm.y1.value&"</y1value>"
xml = xml& "<x2value>"& document.rsqfrm.x2.value&"</x2value>"
xml = xml& "<y2value>"& document.rsqfrm.y2.value&"</y2value>"
xml = xml& "<zonevalue>"& document.rsqfrm.zn.value&"</zonevalue>"
xml = xml& "</sizeofwindow>"
```

The next step is to create an XMLHTTP object to communicate with the backend HTTP server, where the server script for processing the request can be reached. This will enable client-side to pass an XML packet to server using the HTTP connection between the client and the server [5]. The XMLHTTP object can request for opening a connection with an HTTP server, specify the HTTP request method, such as GET or POST, look for the resources such as an ASP script (spatialsearch.asp) for processing the request, and the asysnchronization flag, and finally send the XML packet out to the server using the XMLHTTP send method. Step two of the diagram shows this process. An instance of the corresponding function that is used for initializing and doing tasks can be seen in the following:

```
set xmlhttp=createobject("Microsoft.xmlhttp")
xmlhttp.open "post", "http://localhost/spatialsearch.asp", false
xmlhttp.send xml
```

SERVER SIDE IMPLEMENTATION:

In the server-side, a function is responsible to retrieve the sent client-side XML packet. To do this, there is a need to create a DOM instance and load it with the incoming XML content. The following function is programmed for this process:

```
set xmlRsq= server.CreateObject("Microsoft.xmldom")
xmlRsq.async=false
xmlRsq.load Request
```

After the loading is completed, a DOM tree of the request at the server-side is created. Now, it is possible using DOM programming techniques to extract content of request to further process. Created DOM tree in this phase is used by server-side search function in order to locate requested information. This research, making use of an XML/GML document called metadata.xml, contains registered sensors metadata. This document is used by search function to locate the requested sensor metadata. The following illustrates mechanism followed to access and use of XML document to search and generate the response for the client-side within search function.

1. Loading Metadata.xml as a DOM tree (step 6 of the diagram). To access the information enclosed in the XML/GML document, metadata.xml, the first task is to create a DOM object and then load it with data from the document. What follows shows the relevant function to accomplish this task:

```
Function initialitizedomobject()
dim metadoc
set metadoc=server.CreateObject("microsoft.xmldom")
metadoc.async=false
metadoc.load(server.mapPath("metadata.xml"))
If metadoc.parseError then
 response.write "Error"
Else
 set initialitizedomobject=metadoc.documentElement
End if
End Function
```

2. Creating a DOM tree for the response document (step 7). As discussed earlier, the response of request is sent as an XML packet. To this end, there is a need for a DOM tree to store the response. This can be accomplished by the following function:

```
set spatialsearchRes=server.createobject("Microsoft.xmldom")
spatialsearchRes.async=false
set newnode=spatialsearchRes.createElement("Results")
spatialsearchRes.appendchild(newnode)
```

As the codes above show, the created DOM tree contains a root element named <Results>. This is done by using createElement and appenchild methods.

3. Query for matching the request to the possible information in the XML/GML document (step 5 of the diagram). Here, the search method is based on numeric node index that finds all children of a node and then looks for the interested information within the node. The following is an example to find sensors metadata located in a certain area. The request form for this information, as discussed already, consists of coordinates of the area.

Here, a 'For' loop is used to access the elements of XML document. Then, using a conditional statement, 'If', the nodes of XML document are verified totally to find the requested information.

```
For i=0 to n_member-1
If cdbl(xha) <= cdbl(x2) and cdbl(yha) <= cdbl(y2) and cdbl(xha) >= cdbl(x1) and cdbl(yha) >= cdbl(y1) and zone=zonefile then

NameofSensor=root.childnodes(i).childnodes(0).getElementsByTagName("gml:name").item(0).text
xURL=root.childnodes(i).childnodes(0).getElementsByTagName("accesspoint").item(0).text
xlocation=root.childnodes(i).childnodes(0).childnodes(2).getElementsByTagName("gml:coordinates").item(0).text
.
.
.
End if
Next
```

Provided that the search function finds nodes matching the request, it will set variables for the content of the elements. For instance, in the code above, the 'Name of Sensor' variable is used to store the content of sensor name element. The created variables in this process will be used in the response creation stage.

4. Generating a Response to the client. As mentioned before, a DOM tree is dedicated to store the response. For this purpose, it is needed to create elements required to encode the response within the DOM tree. The following shows part of the function to generate sub elements that are components of response.

```
Set NameSensor= spatialsearchRes.createElement("NameofSensor")
spatialsearchRes.lastchild.lastchild.appendChild(NameSensor)
spatialsearchRes.lastchild.lastchild.lastchild.text = NameofSensor

Set desc = spatialsearchRes.createElement("description")
spatialsearchRes.lastchild.lastchild.appendChild(desc)
spatialsearchRes.lastChild.lastchild.lastchild.text = description

Set zoneasp = spatialsearchRes.createElement("Zone")
spatialsearchRes.lastchild.lastchild.appendChild(zoneasp)
spatialsearchRes.lastChild.lastchild.lastchild.text = zonef

Set location = spatialsearchRes.createElement("SensorLocation")
spatialsearchRes.lastchild.lastchild.appendChild(location)
```

```
spatialsearchRes.lastChild.lastchild.lastchild.text = xlocation
Set unit = spatialsearchRes.createElement("CoordinateSystem")
spatialsearchRes.lastchild.lastchild.appendChild(unit)
spatialsearchRes.lastChild.lastchild.lastchild.text = unitofcoord
```

Previous code shows five new elements built within spatialsearchRes DOM object and determined values for them. As the list shows, these new elements are used to carry Name of Sensor, Description about Sensor, Zone number, Location of Sensor, and name of used map projection (unit).

5. After search function performed its responsibility, the last step in the server-side is to send the response of request toward client-side script (step 8). This task is accomplished using save method of XML DOM that saves and sends the response to the client-side. The following shows the corresponding function to do so:

```
spatialsearchRes.save (response)
```

As mentioned previously, XMLHTTP object on the client-side is responsible for retrieving the response sent by HTTP server.

Till now, it was described how the server-side script manipulates the request and sends the response to the client -side. The next section will discuss, in detail, the client-side and will explain the mechanisms to retrieve and display the sent response by server-side to the user.

As Figure 5 shows, the client-side XMLHTTP (through send method) is in charge of retrieving the response sent by server-side script. To this end, XMLHTTP object supports a property called responseXML that carries the response sent by the HTTP server. The following function is used to create a DOM tree and load the response from responseXML of XMLHTTP object by the client-side script (step 9 of the diagram).

```
set receivedDoc=createobject("Microsoft.xmldom")
receivedDoc.async=false
set receivedDoc =xmlhttp.responseXML
```

As the code above shows, response is loaded into an object called receivedDoc that can be used by the client-side script to further the processing (step 10). The rest of scenario in the client-side includes dealing with receivedDoc object to present the response to the information requester.

To this end, client-side script makes use of a loop structure and node indexes to present the results of request to the user (step 11). The following is an instance function used for this purpose.

```
Function displayresponse (byRef receivedDoc)
If receivedDoc.documentElement.haschildnodes() then
document.write "<h1 align=center>Sensor Metadata Prototype<br>Result of Spatial
```

```
Query (by Window)<br>"
x1=receivedDoc.documentElement.childnodes(0).childnodes(0).text
y1=receivedDoc.documentElement.childnodes(0).childnodes(1).text
x2=receivedDoc.documentElement.childnodes(0).childnodes(2).text
y2=receivedDoc.documentElement.childnodes(0).childnodes(3).text
document.write "Window Size: "&x1&","&y1&" - "&x2&","&y2&"</h1><hr color=green><h2>"
cn=0
For i=0 to receivedDoc.documentElement.childnodes.length-1
cn=cn+1
For j=4 to receivedDoc.documentElement.childnodes(i).childnodes.length-1
if receivedDoc.documentElement.childnodes(i).childnodes(j).nodename="AccessPoint" then
url=receivedDoc.documentElement.childnodes(i).childnodes(j).text
document.write "Access Point: "&"<font color=red><a href='url'> "&url&"</a ><font color=black><p>"
Else
document.write receivedDoc.documentElement.childnodes(i).childnodes(j).nodename &": "& receivedDoc.documentElement.childnodes(i).childnodes(j).text&"<br>"
End if
Next
document.write "<hr color=red>"
Next
document.write "Found Sensor/s: "&cn
Else
msgbox "Sorry, There is no information for entered size of window"
exit function
End if
End function
```

RESULTS

The end result of design and implementation steps is a web-based application that provides facility to do search and access to sensor related metadata. The following figure shows the begging interface of the application when relevant URL of the application is typed in the address bar of a browser.

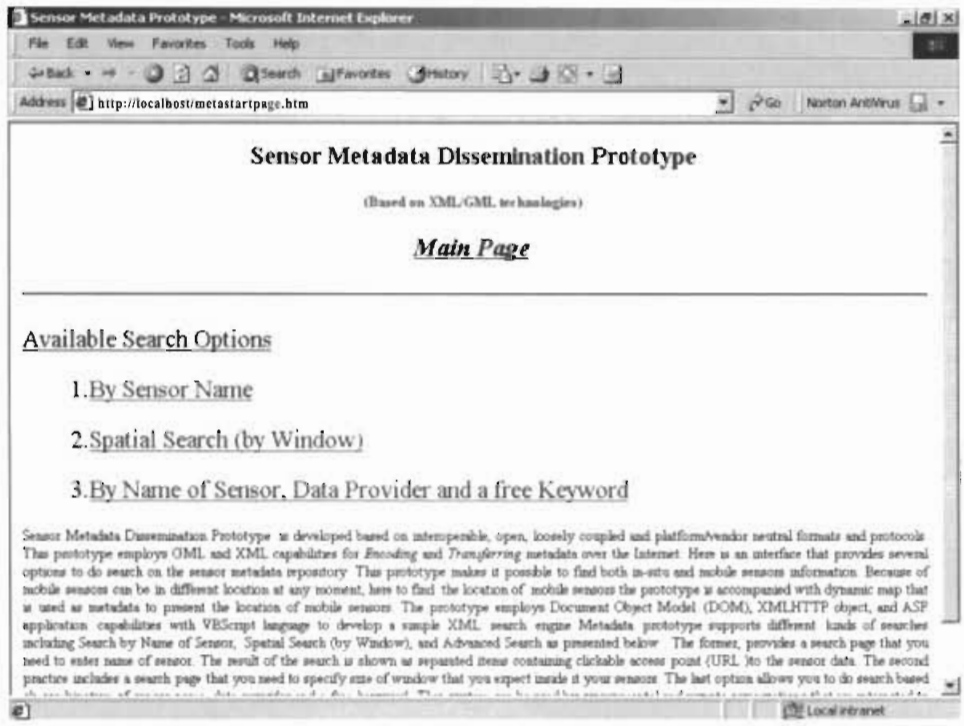

Figure 9: Main page of sensor metadata dissemination system.

This interface provides three options and some guidance for users. The following is devoted to explain functionalities of the available options and relevant interfaces in detail.

SENSOR NAME OPTION

This option allows a user to search based on the name of sensor of interest. Clicking on this option, loads By Sensor Name search interface as shown in Figure 10.

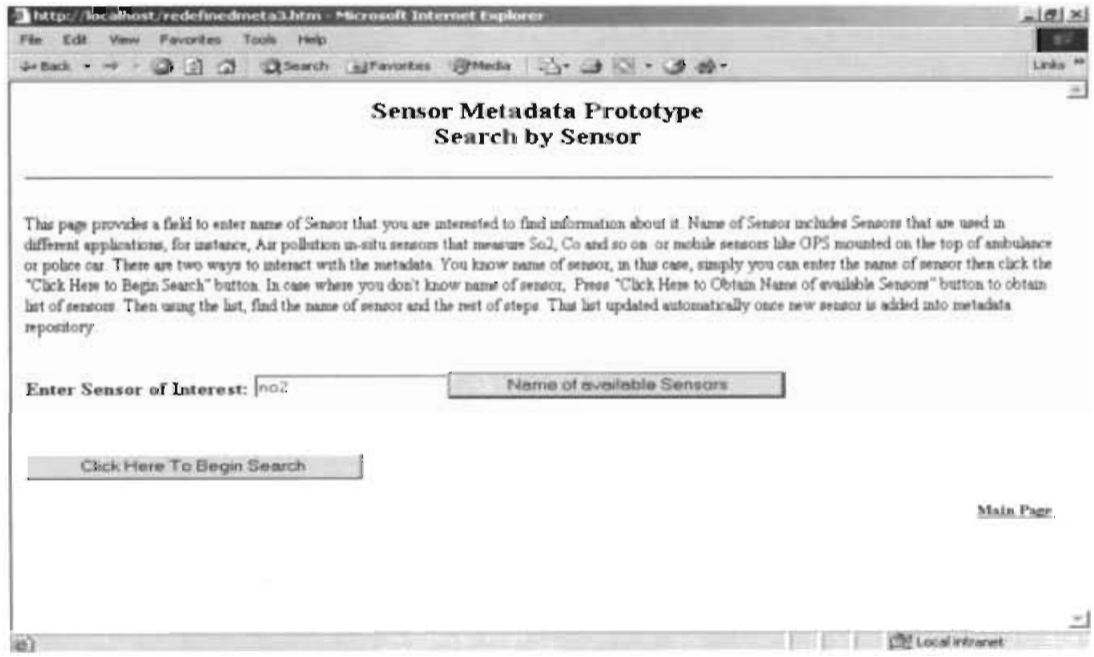

Figure 10: Search by sensor name page of sensor metadata dissemination prototype.

As Figure 10 shows, Search by Sensor page contains a text field that accepts user input. User input should be a sensor name and in case users are interested to get information about available sensor in repository, they can click on 'Name of available Sensors' button. In this case, the system provides a list of registered sensor names for the user. This interface is sensitive to empty field input. In case a user submits empty field, the system will show an error message.

As an example, entering a sensor name in the 'Enter Sensor of Interest' field, for instance No2, and clicking 'Click Here to Begin Search' button will generate output as shown in Figure 11.

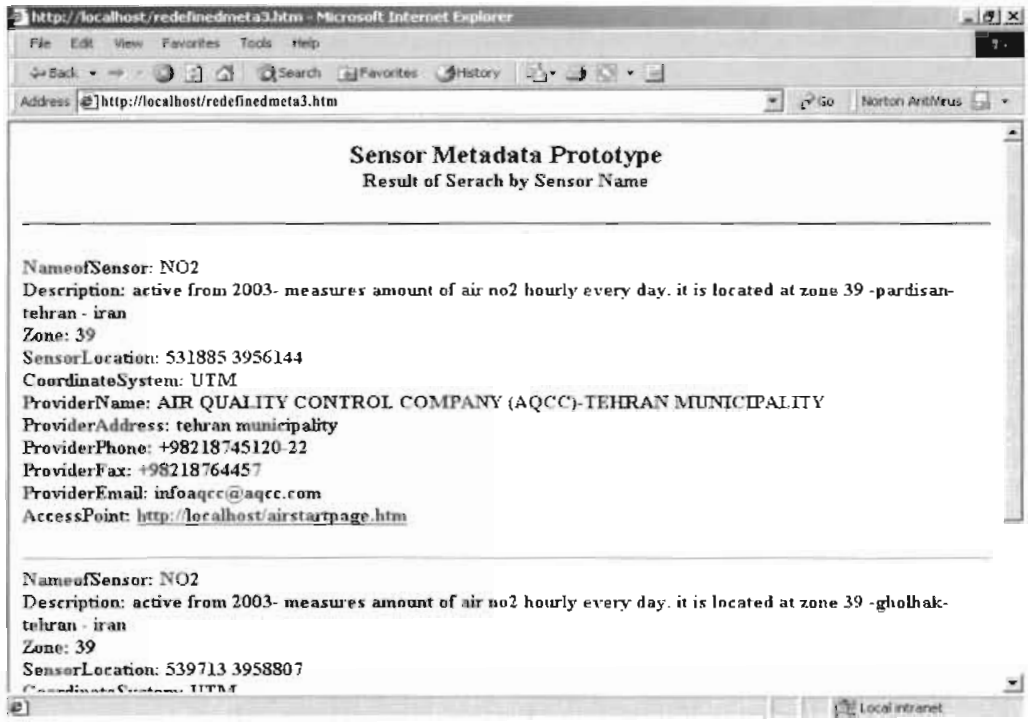

Figure 11: Output of sensor metadata dissemination prototype according to Figure 10 data entry.

Figure 11 shows result of system containing page title and finds No2 air sensors separated by horizontal lines. The following is sensor metadata elements reported by the system:

1. Name of Sensor, which points to sensor name.
2. Description, which gives more information about sensor dataset.
3. Zone, Which refers to sensor location UTM zone number.
4. Sensor Location, which points to geographic position of sensor.
5. Coordinate System, which points to the name of used map projection (UTM).
6. Provider Name, which points to the name of sensor data provider. It can be name of a company or person.
7. Provider Address.

8. Provider Phone.
9. Provider Fax.
10. Provider Email.
11. Access Point (URL), which provides a link to access to sensor data set.

SPATIAL SEARCH OPTION

Spatial Search option makes it possible to find information regarding deployed sensors within a specified area. This area is specified using its coordinates. Clicking on this option loads the relevant interface shown in Figure 12. This interface contains five fields to specify coordinates (window) and UTM zone number of the area.

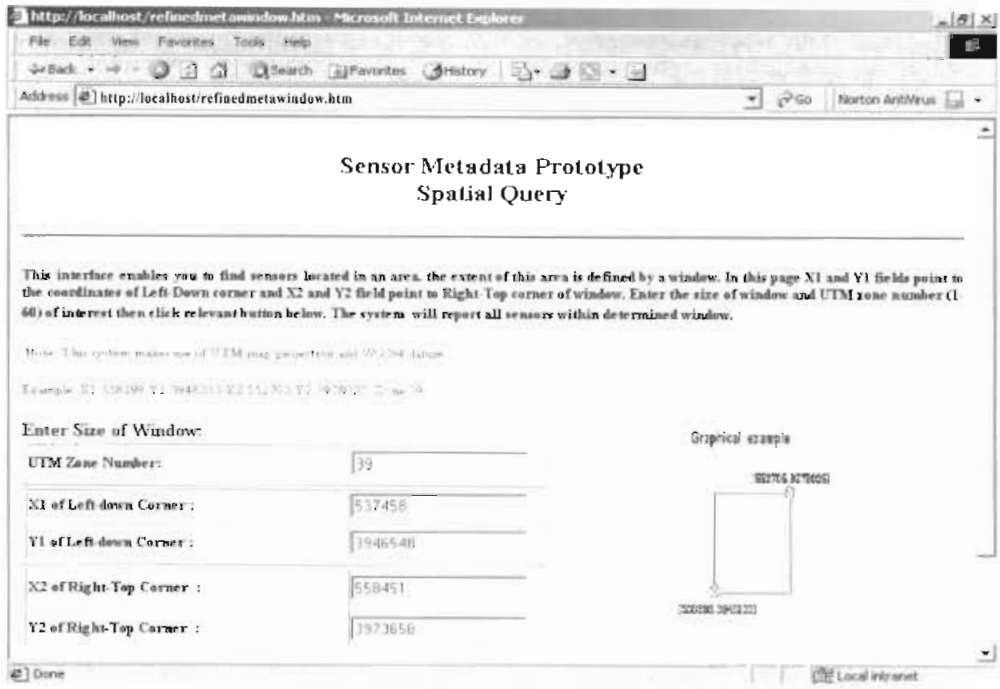

Figure 12: Spatial query page of sensor metadata dissemination prototype.

This page also provides guidance for user regarding how to determine size of window (by text and picture). This interface checks user inputs values before sending them toward server-side script. The following includes system restrictions when data entering:

- "UTM zone number" field value must be positive, at most 2 digits, and numeric.
- "X1 and X2 of Left down/top corner" fields value must be positive, 6 digits and numeric.
- "Y1 and Y2 of Left down/top corner" fields value must be positive, 7 digits and numeric.

In case the user pays no attention to the mentioned restrictions, the system will produce an error message and does not allow continuing the work. Figure 12 shows user input in which zone number is 39 and specifications of area are defined as (537458, 3946548) -

(558451, 3973658). In this example, first point refers to coordinates of left-down corner of window and second one refers to right-top. The following figure shows output for the mentioned user input.

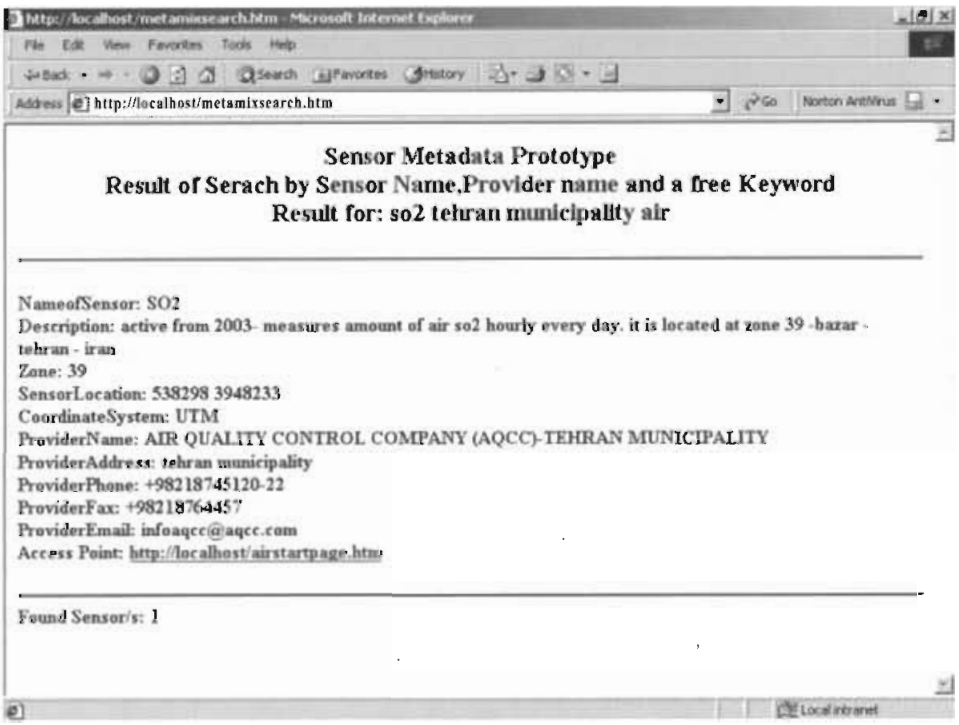

Figure 13: A part of output of sensor metadata dissemination prototype according to Figure 12 data entry.

As Figure 13 shows, this output includes found sensors along with their relevant metadata within determined area that are separated by horizontal lines. Due to space limitation, Figure 13 only shows Co sensor and relevant metadata completely.

NAME OF SENSOR, DATA PROVIDER AND A FREE KEYWORD (ADVANCED SEARCH)) OPTION

This option allows the users to search, based on the combination of three parameters namely sensor name, provider name, and a free keyword of interest. Clicking on this option loads the interface shown in Figure 14. The free keyword in this interface can be anything that helps the user to reach the information of interest. For instance, it can be "air" or "water" that point to environments that sensors can be deployed. This interface also provides facilities to users to reach the list of registered sensors and data provider names. Two button 'Name of available sensors' and 'Name of available providers' can be used for these purposes. Figure 14 also shows an example that makes use of combination of So2 as sensor name, Tehran municipality as data provider name and air as keyword in order to find sensors metadata based on these information.

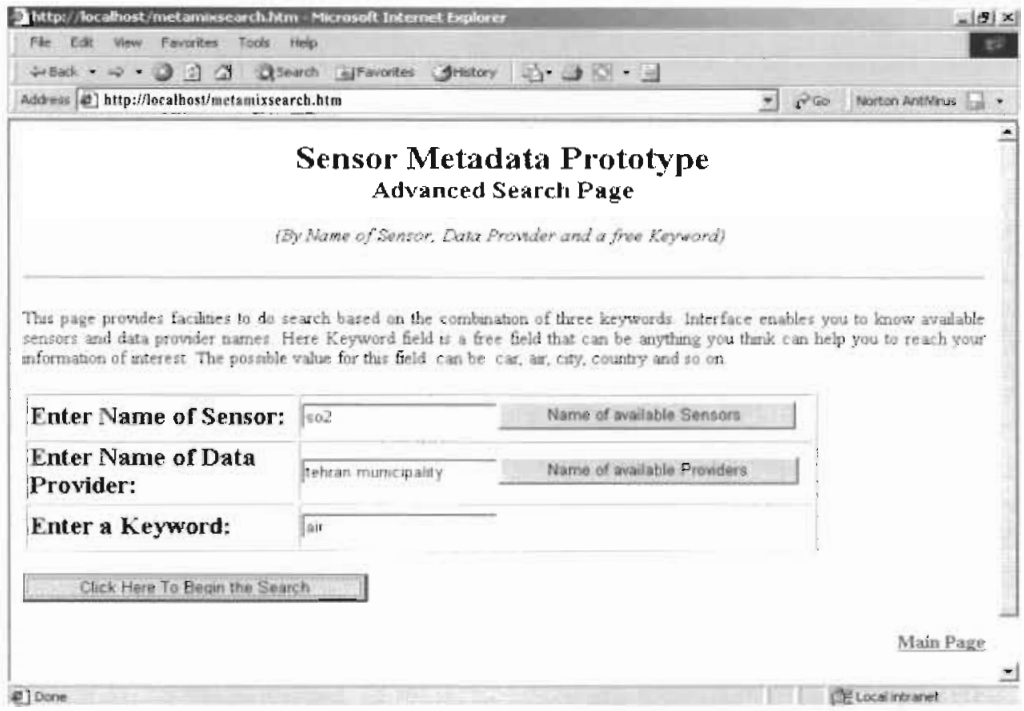

Figure 14: The advanced Search page of sensor metadata dissemination prototype.

Submitting user input corresponding with Figure 14 will generate output shown in the following figure:

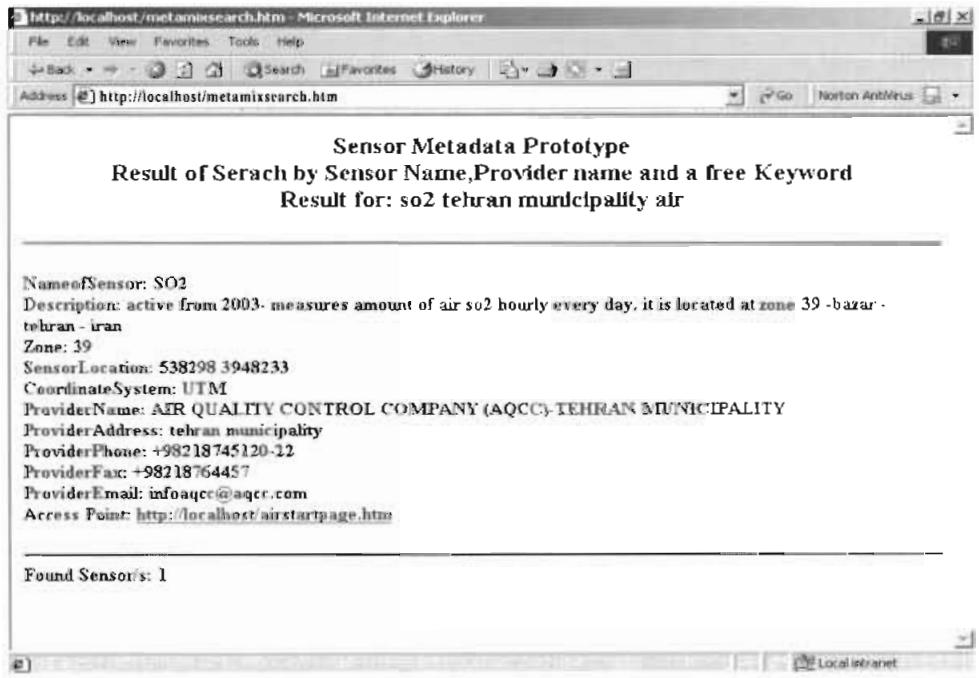

Figure 15: Output of sensor metadata dissemination prototype according to Figure 14 data entry.

As the above figure shows, there is only one sensor in the repository at the search time, which matches the request. Based on this output, So2 is the found sensor that is

located in UTM zone number 39, at location 538298, 3948233 and relevant dataset for this sensor is available in the URL specified in the Access Point field. In addition, specifications of Tehran municipality as data provider can be seen in the output.

CONCLUSIONS

This research demonstrated an application of XML and its derivative GML in building a sensor metadata repository. It was found that XML and GML are very powerful tools to model metadata file for sensors. The use of XML enforces XML features like interoperability; openness, flexibility and loose coupling on metadata file in this system and on the other hand, validation rules improve the reliance of the information stored in the sensor metadata repository. But it is realized that the use of XML/GML based documents may result in huge documents, reducing data processing speed on the web and network environments. Moreover, use of Scalable Vector Graphic (SVG) technology along with GML will improve functionality of developed application in this research. Therefore, this issue is recommended as a topic for further research for those who are interested in Web GIS related studies.

REFERENCES

[1] Christoph, W. and Koller, C. "Active Server Pages in 24 Hours." *Sams Publishing*, pp. 381-396, 2002.

[2] Groot, R. and Mclaughlim, J. "Geospatial Data Infrastructure." *Oxford University Press*, pp. 25-28, 1999.

[3] ITC, "Principles of Geographic Information Systems." *ITC publishing*, pp. 206-208, 2001.

[4] Larry, k., "The Official XMLSPY Handbook." *Wiley publishing*, pp. 55-85, 2003.

[5] MeeFoo, S. and MengLee, W. "XML Programming Using the Microsoft XML Parser." *Apress Publishing*, pp. 35-39, 195-245, 2002.

[6] Open GIS Consortium (OGC) Inc, "OpenGIS® Geography Markup Language (GML) Implementation Specification." Available: at <http://member.opengis.org/tc/archives/02-023r4.pdf>, 2003.

[7] Open GIS Consortium (OGC) Inc, Overview of OpenGIS Implementation Specifications. Available at: <http://www.opengis.org/info/gisworld/200110911.TS.SpecOver.htm>, 2002

[8] Schmuller, J., "SAMS Teach Yourself UML in 24 Hours." *Sams Publishing*, pp. 30-36, 1999.

[9] Vlist, E., "XML Schema." *Oreilly*, pp. 6-12, .2002.