

An Implementation Model of Open Distributed Object GIS for Multiple GIS Server based on CORBA

Heui-chae Jin¹ Jin-Rak Son²

¹ *Informatization Evaluation & Analysis Division, National Computerization Agency*
168, Jukjun-Ri, Suji-Eub, Yongin-City, Gyunggi-Do, Rep. Of Korea, 449-717
Tel: 82-331-2602433, Fax: 82-331-2602426
E-mail: hcjin@nca.or.kr

² *Center for National Informatization, National Computerization Agency*
168, Jukjun-Ri, Suji-Eub, Yongin-City, Gyunggi-Do, Rep. Of Korea, 449-717
Tel: 82-331-2602317, Fax: 82-331-2622753
E-mail: jrson@nca.or.kr

ABSTRACT As the Information technologies evolute and the system integration is implemented widely, we need to share the geospatial information in order to diffuse the usability of GIS. The method to share the geospatial information may be classified into two types. The one is Transfer method that we transfer the spatial data to another system in itself, the other is Access method that a user directly access to our system to get the geospatial data by the distributed process in a local remote machine. Actually, Access method is more reasonable than Transfer method because the network has not enough bandwidth to frequently transfer the huge geospatial data at a time. So, the system must be a distributed object system to realize the share of geospatial data and must be open system to assure the interoperability between distributed objects. Also, it is necessary that we may permit many users to access the geospatial data from multiple platform systems to grow up the efficiency of accessibility. In this paper, we present the open distributed object system based on OMG CORBA to offer the data access from multiple GIS server that have the specific geospatial data and have a different kinds of system platform. The open distributed object GIS model consists of the distributed object GIS server and client, access manager based on CORBA to distribute the geospatial service. Then we suggest the method to build-up the system and experiments on the system used by JAVA.

KEY WORDS GIS, Distributed System, Implementation Model, Multiple GIS Server

1. Introduction

Recently, there are widely built up the Geographic Information System(GIS) which controls and erects the electronic Geographic Information(GI) to manage the geographic phenomenon systematically and efficiently. But GIS is one of the system which require so many time and cost to gain the database.

Nevertheless, in general, we built up and use the isolated system with Geographic Information Database(GIDB) because we haven't the model to share or distribute the information between those of GIS. Also we may not have the sufficient interoperability of data among different systems.

It is necessary that we do share smoothly the GI among GIS's to diffuse the GIS application. Methods to share the information are divided by two types. The one is Geographic Information Transfer Method(GITM) that we transfer the spatial data into another system in itself, the other is Geographic Information Access Method(GIAM) that a user directly access to another system to get the geospatial data by the distributed process in a local remote machine. In this study, we take into account at GIAM because GITM is difficult to use the network like internet in consideration of heavy

volume GI. As to realize the efficient sharing of information through GIAM, we needs the Distributed Object System, To get the interoperability between the Distributed Object of GIS, we needs the open system. Also, it is necessary that we approve the access to identical GI in Multiple Servers to access the data efficiently in spite of the heavy volume of GI.

In this study, we suggest the model of distributed object GIS server to supply the GIA Service and access distribution system to dispersion of access to multiple GIS server. Also we have implementation the system included distributed object GIS client with JAVA based on CORBA of OMG and multiple GIS server with identical GI.

2. Geographic Information Sharing

As the geographic Information technologies evolute and the data is accumulated, we are used the geospatial information system in various area. So, we more enlarge the importance and necessity of sharing the GI between systems, like topographical map, underground facility map, road map, administrative division map, urban planning map.

For sharing of GI, we can reduce the cost of setup for GIS and use the exact information relative to geo-space. Now, we go into details about methods for sharing of GI by each types.

2.1. Geographic Information Transfer Method(GITM)

GITM is the methods that download GI directly from GIS server(s), store GI in local GIS and practical use at client.

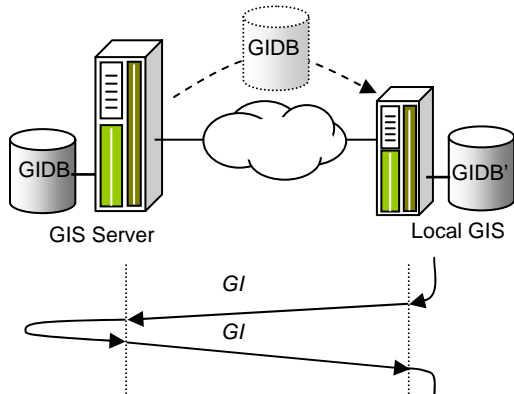


Fig.1 Geographic Information Transfer

Fig. 1 represents the procedure how to use the GI at client. If an user of the local GIS request GI to GIS server(s), GIS server provide the requested GI to the user of local GIS. Then the user can use the provided GI in a local GIS.

GITM can improve the performance of various processes by means of transmitting the GI stored in GIS serve to a local GIS. But it need so many time to transfer the GI from server to local GIS initially and additional effort to maintain the data integrity of GI transferred to local GIS from GIS server.

2.2. Geographic Information Access Method(GIAM)

GIAM is the method that an user of local GIS access and use the GI of GIS server(s) directly. This is the same method of traditional client-server.

Fig. 2 represents the procedure how to use the GI at client in GIAM. First, an user of local GIS request the service related GI to GIS server(s). Then GIS server executes the service of requested GI and transmit the result to the user of the local GIS. Namely, Though the client has not GI database in a local GIS, the user can do the process or service with GIDB in GIS server.

GIAM reduce the initial time to gain the data from GIS server owing not to transmit the GI from GIS server to local GIS of user in a different way of GITM. So we can build the local GIS in a small size because local GIS may not have the GIDB to use application

But, this method requires the continuous network connection between local GIS and GIS server. So in the case of small transaction frequently or transaction of very large volume of GI, there may drop down the performance of the whole system.

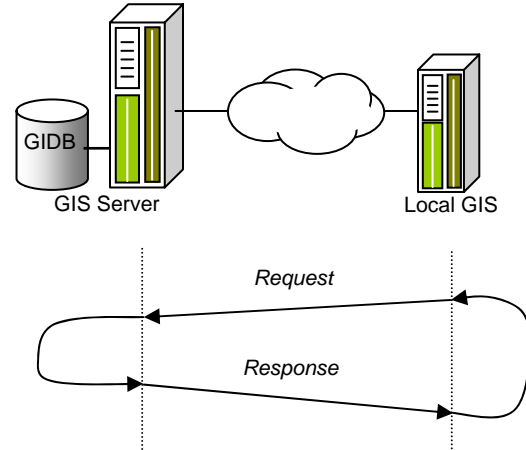


Fig.2 Geographic Information Access

3. CORBA

CORBA(Common Object Request Broker Architecture) provides a specification for the object-oriented distributed system in a language, operating system, platform and vendor independent way.

OMG developed the standard framework 'Open Management Architecture(OMA)' which can integrate the application based on object-oriented technology in the heterogeneous distributed environment. OMA is composed of object model and reference model. Core object model can represent simply about concepts and terms of the object-oriented database, User Interface(IU) and definition extensively with respect to various fields.

Reference model is the software component environment suggested by OMG and is composed of 4 following components. system. Also, it is not necessary that another process must be required to maintain the integrity of GI between GIS server and local GIS.

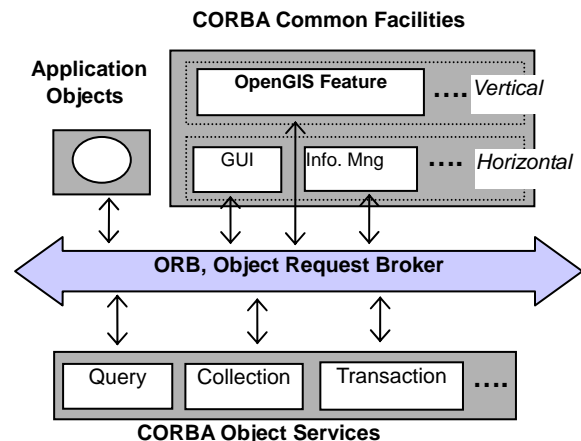


Fig.3 OMA Reference Model

3.1. Object Request Broker, ORB

ORB is the core object middleware to make relationship between distributed objects which composed of mechanisms and interfaces to request and response with each other.

3.2. CORBA Object Service

CORBA object services provide the many functions to treat objects with respect to the system. Additional announces to reinforce the ORB functions of the system services like security, database interconnection are in progress.

3.3. CORBA Common Facilities

CORBA common facilities are the set of components to provide the direct services to the application object of domain dependent level, for example, finance and medical treatment.

3.4. Application Objects

Application object is the end-user application.

Especially, ORB is the core function in CORBA. In spite of ORB objects is distributed in somewhere of the network, it may call the method and return the response of the result from the method. CORBA is the specification of the ORB common architecture, or distributed object middleware.

access the CORBA Objects through the application object, we must get the IOR(Interoperable Object Reference) of objects, previously. Application object may call the method of object implementation by IOR. Merely, client is the program entity which call the operation of object implementation. Client put to practical use the static call method(applied IDL Stub) or dynamic call method (applied Dynamic Invocation Interface : DII). Also client uses interfaces provided by ORM directly. ORB can be transmitted by parameter, control can be transmitted by (Static) IDL skeleton and dynamic skeleton to object implementation. When the client do the request, object implementation receive the service of ORB by Object Adapter, deliver the control and result to client after doing the request.

4. Open GIS Specification

CORBA Implementation Specification is the set of interfaces to build the distributed object of GIS based on OMG's CORBA. It is envisaged that this specification will become a candidate for inclusion in the OMG's work as a vertical CORBA facility covering geospatial information management. CORBA Implementation Specification has defined standard interfaces related to feature, geometry and SRS(Spatial Reference System) by CORBA IDL (Interface Definition Language). This specification, in the point of implementation view, suggest various adaptability like transformed DB technique(i.e., Object Wrapping) with respect to GI stored in traditional RDBM and can be built a system of programming language, operating system independent way.

4.1 Feature Module

The structure of feature model suggested by abstract specification of OGC is the same like Fig. 5.

Feature module provides function to create, access, query related to geospatial features and feature Collection. System may enable client applications to create new Feature objects through a FeatureFactory object. Features have a set of properties, which may be iterated over by a FeatureProperty Iterator. Feature can create the implemented object by FeaturePropertyIterator, get the reference and can access properties of an implemented object of feature through the reference.

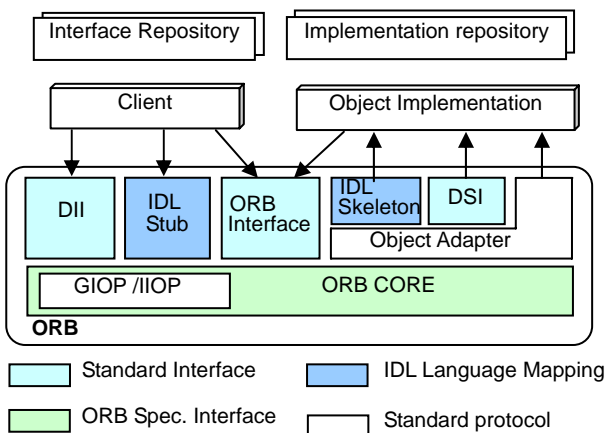
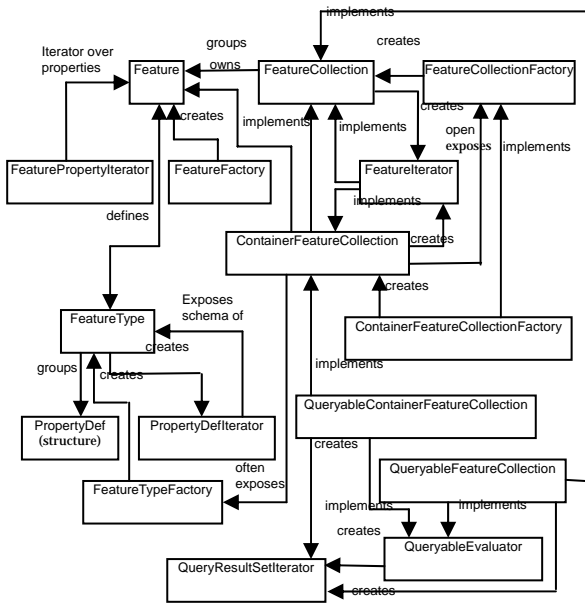


Fig.4 CORBA ORB Architecture

CORBA apply the concept of object reference to communicate between the objects. When we

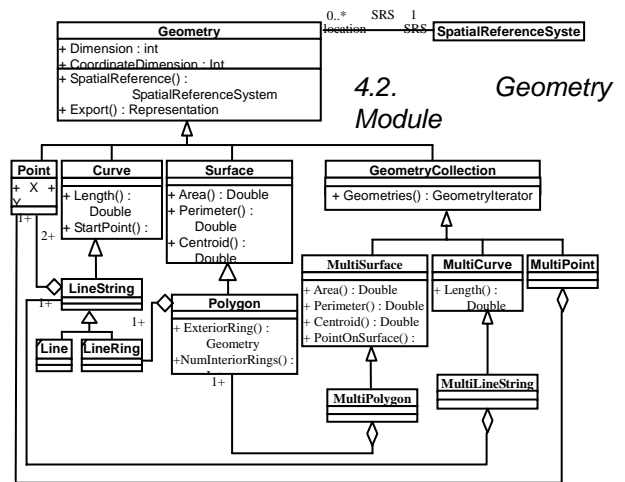


Feature Collection is a group of features and can access for iterating over Feature included in Feature Collection through Feature Iterator. There are another interfaces details of Feature Collection (i.e., Container Feature Collection interface and Queryable Container Feature Collection) to represent Feature set having different properties.

Feature Collection has the reference of Feature, not contain the actual Feature object. In general, Feature Collections only refer to their member Features. But Container Feature Collection has the actual entity of Feature. So if we are remove the Feature in Container Feature Collection, we are aware that the Feature disappeared immediately. Namely, ContainerFeatureCollection is the same role of database in GIS. Generally, client can access to the Feature module through Container Feature Collection. Queryable Container Feature Collection interface provides support for expressing and evaluating queries against a container collection's contents. The query languages supported, as well as the iteration over the query result set, are inherited from the ContainerFeatureCollection.

In OpenGIS, we can get the result set of queries through Queryable Result Set Iterator. Feature Collection is created through Feature Collection Factory, ContainerFeatureCollection is created through ContainerFeatureCollectionFactory, Queryable Container FeatureCollection is created through QueryableContainerFeatureCollectionFactory.

Feature is composed of spatial data represented by geometry and the set of non-spatial data. In Fig. 6, Property Set is composed of geometric property and property. Geometric property is linked with coordinate geometry and has non-geometric data in the form of ANY type of CORBA. This Property Set is the same as database. The subset or entire set of PropertySet become the Feature.



Geometry module provides services to create, access and operate the geospatial function. Geometry module is divided into two types. The one is the set of interfaces that represents the geospatial data itself. The other is the set of interfaces that represents the coordinate system with respect to the geospatial data.

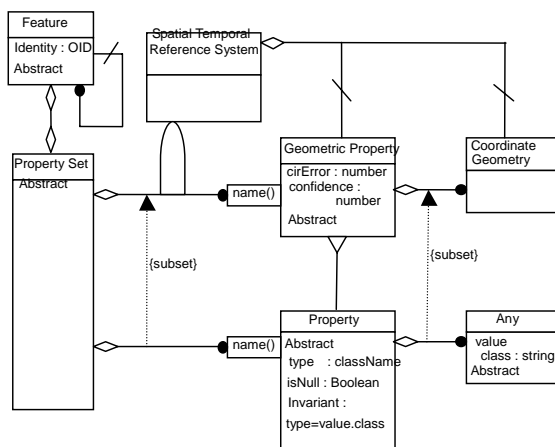


Fig.6 Relationship between Feature and Geometry

All Interfaces to represent the geospatial data

are inherited from geometry. Geometry has the general properties of geospatial data and define the spatial operation with respect to geospatial data. Geospatial data is represented with various interfaces with respect to dimensions of the geometry. Geometries are categorized by their dimension as 0-dimensional geometries (points), 1-dimensional geometries (curves), 2-dimensional geometries (surfaces), 3-dimensional geometries (solids) and 4-dimensional geometries (hyper-solids). (This is only an classification, not mean the general dimension)

Interfaces to represent the Coordinate system are Geographic Coordinate System and Projected Coordinate System inherited from Spatial Reference System. Geographic Coordinate System is mapping the surface of earth to ellipsoid and Projected Coordinate System is mapping the surface of earth to a plane through projection.

5. An Implementation Model of Open Distributed Object GIS

5.1. Implementation Model

In CORBA environment, we may have connection between client and server of GIS through OpenGIS standard interfaces. Let's suppose that GIS client wants to call a certain interface among OpenGIS standard interfaces. First we must get the reference of the implemented object of the interface from CORBA Naming Service. When we call the method of the interface through the reference, GIS server which has the requested method call the operable object and return the result to GIS client.

At this time, we may use the Container Feature Collection interface or Queryable Container Feature Collection interface to access and query to GI in GIS DB, the __Factory series interfaces(i.e, Feature Type factory, Container Feature Factory) through reference of interface to create, delete of class or object in GIS DB.

With respect to each interface in GIS server, the object accesses to DB, retrieves the GI from DB through access interfaces provided by their DBMS. The result of retrieval is stored in GIS server or returned the result to client after transformed to the reply type of interfaces defined by OpenGIS. In case of storing the result in server, server creates the available object of __Iterator series to access the result of retrieval one by one and return the reference to GIS client. GIS client can access the result stored in GIS server by means of returned reference.

In CORBA environment, there are two methods that client find the object reference in servers - static method and dynamic method. Static method

transmit the message to server by IDL compiled stub, dynamic method transmit the message to server by Dynamic Invocation Interface(DII) of ORB.

Static method has the merit that we develop program easily, are provided stable and rapid services. In dynamic method, we can develop client objects without concerning objects of server. In this study, we are based on the dynamic method.

An Implementation model mentioned above is worked under the conditions of a single client and a single server. That is a 1:1 transaction model based

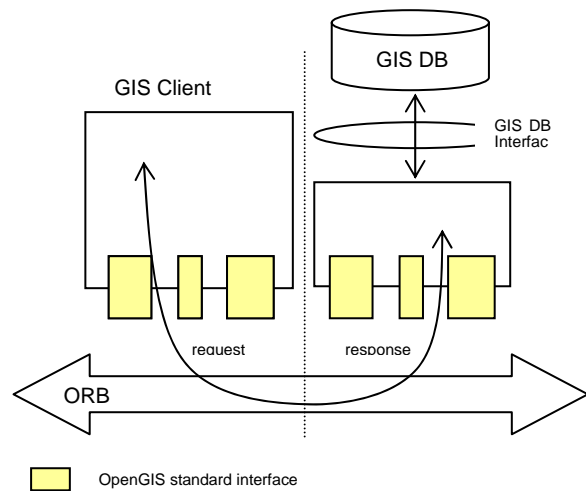


Fig.8 Open Distributed Object GIS Model

the fundamental pattern among CORBA communication type.

But, in the case of server which deal with very heavy data like GIS DB, we must provide the same services in multi-servers to offer a stable services to multi-clients. So we need to develop a model to support this situation.

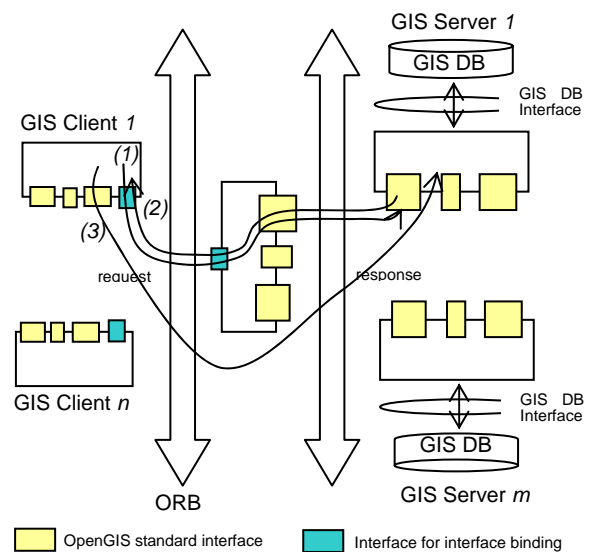


Fig.9 Open Distributed Object GIS Model for multiple server

In the model of Fig. 9, we investigate how GIS client to get the reference of the operable object from anonymous GIS server.

First, a client of GIS accesses the middleware to be provided the distributed object services and request the object reference of the service. The middleware received the request acquire the object reference from CORBA Naming Service and return the reference of requested object through the acquired reference. When the middleware receive the object reference, the middleware deliver it to GIS client. GIS client may directly access the GIS server to do process through the transmitted object reference. It can improve the performance of GIS service because the traffic of accession is dispersed to server by means of middleware.

5.2. Implementation example used JAVA

The following is example of client and server system with Orbix and JAVA.

We investigate how to use the CORBA Naming Service with Orbix in respect of server and client. First, we must acquire the object reference of Naming Service to register the object in server. We use the method "NameService" to acquire the object reference when we call the method, *reolve_initial_reference()* of org.omg.CORBA.ORB class. The *resolve_initial_reference()* return the object reference of Naming Service from the selected server. The server registers the name of the returned object reference in Naming Service and wait for the client's request.

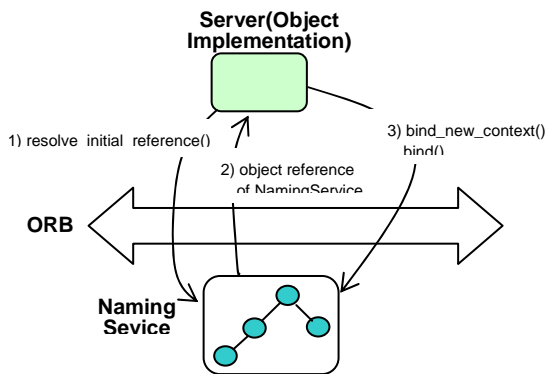


Fig.10 Server operation for Naming Service

Now, we represent above procedures by JAVA pseudo code follows. We example the implementation of the object in server with respect to ContainerFeatureCollection.

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackag-
```

e.*;

```
public class ContainerFeatureCollectionServer{
    ORB orb;
    NameComponent [] name;
    NamingContext rootContext, cfcContext, fcContext;
    try {
        cfc=new ContainerFeatureCollectionImple("cfc");
    } catch {
        .....
    }
    try {
        orb = ORB.init(args,null); // orb initialize
    }

    objRef=orb.resolve_initial_reference("Name-Service");
    rootContext=NamingContextHelper.narrow(obj-Ref);
    // find initial Naming Context
    name = new NameComponent[1];
    name[0] = new NameComponent("CFC", "");
    cfcContext=rootContext.bind_new_context(n-ame);
    name[0] = new NameComponent("FC", "");
    fcContext=cfcContext.bind_new_context(na-me);
    fcContext.bind(name,"cfc");
} catch (NotFound se) {
}
}

class ContainerFeatureCollectionImple
    extends _ContainerFeatureCollectionImplBas-e
{ public String m_name;
    public ContainerFeatureCollectionImple(String name){
        .....
    }
    .....
}
```

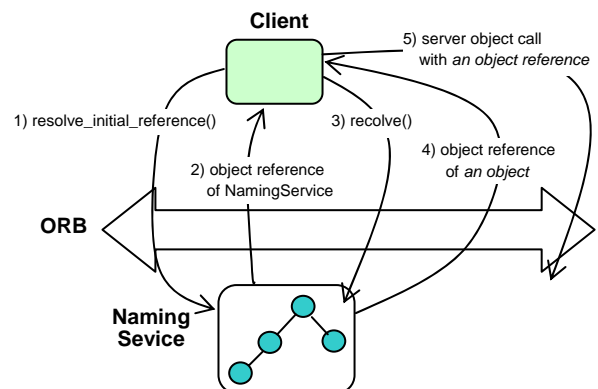


Fig.11 Client operation for Naming Service

As showed JAVA pseudo code, server create the NamingGraph by access the Name Server which have taken charge of the Naming Service. At this time, the bind between object and name be made by *bind()* method or *rebind()* method.

In the case of client, client call *resolve_initial_reference()*, same as server case, to

refer the object of the Naming Service with parameter "NameService". The *resolve_initial_reference()* method transmit the object reference of Naming Service to the client. Client execute the *resolve()* process with the parameter, structure list of Name. And the client is returned the object reference correspond to the needed object from Naming Service. Then, by the returned object reference, the client call the object which is implemented in the server.

In the case of client, Naming Service uses to find the initial context and interpret the name to acquire the object reference. Following is the JAVA code related to ContainerFeatureCollection with respect to client.

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

public class ContainerFeatureCollectionServer{
    ORB orb;
    NameComponent [] name;
    NamingContext rootContext, cfcContext,
    fcContext;try {
        orb = ORB.init(args,null); // orb initialize

    objRef=orb.resolve_initial_reference("NameService");
    rootContext=NamingContextHelper.narrow(objRef);
    // find initial Naming Context
    name = new NameComponent[1];
    name[0] = new NameComponent("CFC","");
    name[1] = new NameComponent("FC","");
    name[3] = new NameComponent("F","");
    objRef = rootContext.resolve(name);

    F=ContainerFeatureCollectionHelper.narrow(objRef);
    .....
    } catch (NotFoundException) {
    }
}
```

Client call the *resolve()* method to find the object by name. The *resolve()* method return the object reference with relate to given name or Naming Context. Return type is IDL Object. Client get the returned value by *org.omg.CORBA.Object* type of JAVA. Therefore, return value must be cut down the extent by narrow method before used by the application.

6. Conclusion

We suggest the implementation model to support the multiple servers in open environment to smoothly sharing the information among GIS's. Also we suggest the way of implementation by CORBA

of OMG, for example.

CORBA can aid the system to build the programs with language independent way and to integrate those with respect to executable code in the environment of heterogeneous systems or different operating system.

This function of CORBA is the same as the interoperability with respect to source code or executable code of JAVA in heterogeneous system.

Also, Applet and Servlet of JAVA are good for integrating with Web. In the consideration of the usability and interoperability of JAVA and CORBA, it is more extended the area of JAVA implementation based on distributed objects GIS with CORBA. Especially, in the case of distributed object GIS client, it is necessary that client system must be JAVA in the consideration of integrating with common UI or Web.

To suggest the distributed object GIS implementation model with interoperable JAVA related to OGC's CORBA specification, we can build the open system to fit the standard. As improved the function of packages, definition of GIS data details, we can acquire the more stable interoperability of the total system.

To improve the performance of access to GI in the consideration of heavy database of GIS, we prepare the permission of the identical access to the geospatial data from multiple systems. We make an effort to implement those subjects through CORBA and JAVA as middleware.

Reference

- OpenGIS Consortium, 1999, OpenGIS Project Document 99-016, Revisions to the Query Model for OpenGIS Simple Features Specification for CORBA, OpenGIS Consortium.
- Open GIS Consortium, 1999, The OpenGIS Abstract Specification Model, Version 4, OpenGIS Consortium.
- Open GIS Consortium, 1999, OpenGIS Simple Features Specification for OLE/ COM Revision 1.1, OpenGIS Consortium.
- Open GIS Consortium, 1999, OpenGIS Simple Features Specification for SQL Revision 1.1, OpenGIS Consortium.
- Open GIS Consortium, 1998, OpenGIS Simple Features Specification for CORBA, Revision 1.0, OpenGIS Consortium.
- Orbix - Reference, IONA
- Orfalic, Robert, 1997, Client/server programming with Java and CORBA, John Wiley & Sons Inc.
- Son jinrak, Jung junWon, Jin Heui-Chae, 1999, A Development of the Distributed Object GIS based on CORBA Implementation Specification with JAVA, Korea OpenGIS Research Society
- Siegel, Jon, 1999, CORBA fundamentals and

progeamming, John Wiley & Sons Inc.

Vogel, Andreas, 1998, Java programming with CORBA, Wiley