A Geographical Information System Based on Agent Architecture *

TANG Chao[1], FENG Shan[1], XU Li-da[2]

[1](*Institute of Systems Engineering, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan* 430074, *China*);

[2](*Department of Management Science and Information System, Wright State University, Dayton, USA*)

E-mail: sfeng@mail. hust. edu. cn

http://www. hust. edu. cn

Received April 24, 2000; accepted November 14, 2000

Abstract: In this paper, Agent-based architecture has been introduced into GIS for interactability and scalability. The authors have developed an interface agent to interactively assist the user with the query formation process. The interface agent has a capability of offering domain knowledge that is associated with a given query. Also the agent records user's troubleshooting experience and shows it to the same user and other user as hints. Further, the authors also have developed the Agent-based architecture with a 2-tier mediator model in GIS to meet the needs for interoperability among diverse application domains and integration of heterogeneous information source. The broker agent provides a matchmaking service to other agent. As a result, qualitative and quantitative data can be integrated into GIS. Finally a prototype system——GXGIS, using agent-based technology is described.

Key words: interface agent; geographical information system; knowledge base; heterogeneous information source; mediator

Geographical Information Systems (GIS) have gained popularity in recent years. GIS software is unique in its ability to capture, store, and manage spatially referenced data such as points, lines, and polygons (vector data model), or as continuous fields (raster data model). A conventional GIS incorporates digitized maps with database systems. In particular, it is useful in a wide range of applications dealing with geographical and topological data, such as logistical planning, oil/mineral exploration, environmental monitoring, archeology, epidemiology, and so on. For many application domains, there is a need to extend the GIS environment to include several information sources. These information sources, such as object-oriented database management system, relation database management system and files system, run on network infrastructures to support certain users in their domain.

Early research on GIS is primarily concerned with the methodologies of efficient integration of maps and

* Supported by the National Natural Science Foundation of China under Grant No. 79990850 (国家自然科学基金)

TANG Chao was born in 1974. He is a Ph. D. candidate at the Department of Automatic Control Engineering, Huazhong University of Science and Technology. His research interests are Geographical Information Systems, multi-agent system. **FENG Shan** was born in 1935. She is the director of the Research Division on Intelligence Integrated Systems. Her current research interests are agent oriented programming, and application of distributed Artificial Intelligence. **XU Li-da** was born in 1949. He is a professor at the Department of Management Science and Information System, Wright State University, USA. His current research field is artificial intelligence and its applications. He is a guest professor of Huazhong University of Science and Technology.

databases. Map-based spatial operators are later introduced to improve the query answering capabilities. Such operators are area, length, centroid, perimeter, located, within, contain, union, intersection, and difference[1,2]. In addition, approximate spatial operators, such as near-to, between, and south-of, are also studied and implemented in some systems[2]. The advent of knowledge based systems promises many new ways to enhance GIS.

Fundamentally the end-users, regardless of emphasis, share one characteristic that they use GIS as a problem analysis and solving tool. The production of cartographic quality maps and presentations can certainly provide support in many pertinent decision processes. But in a large information system, the size and complexity of the schema often make it difficult for the user to pose a query. This motivated us to develop a knowledge-based agent to interactively assist the user with the query formation process. Also the integration of the geographical/spatial and other information sources in GIS greatly facilitates the information retrieval and analysis. We provide an agent-broker architecture to integrate and access the heterogeneous information sources.

The remainder of the paper is organized as follows. First, we describe an implementation of GIS in Section 1. Then the implementation of interface agent and how it works are given in Section 2. Section 3 shows the existing technology to access and integrate heterogeneous information sources. Section 4 presents an agent-broker architecture for integration of heterogeneous information source using a 2-tier mediator model and the integrated view. Finally, Section 5 discusses the benefits of our agent-based GIS.

1 Geographical Information System

Maps are simplification of the real world and are, therefore, model of reality. GIS can be viewed in this way. Geographically, the system is concerned with data related to geography and geographic scale of measurement. The system allows for the storage and extraction specific and meaningful attribute information. These data are connected to some geography, and are organized around a model of real world. And, an automated system should include an integrated set of data and procedure.

A geographical information system allows a user to pose queries based on geographical objects such as countries, states, cities, airports, rivers, etc. Further, it provides spatial operators for constraining the geographical queries. In this section, we discuss the basic elements of a geographical information system.

1.1 Geographical objects and representations

There are three types of geographical objects: points, lines, and areas. For example, cities, schools, or airports can be shown as points on a map. Highways, streets, or rivers can be lines; countries, states, or lakes can be areas. A point, which is a single $(X;Y)$ coordinate, represents a geographic feature too small to be depicted as a line or an area. A line is a set of ordered coordinates that represent the shape of a geographic feature too narrow to be displayed as an area. An area feature is a closed figure (series of arcs comprising its boundary) enclosing a homogeneous area, such as a state or water body. These objects can be represented in one of the two forms:

1. Raster: raster representations use a set of small and regular grids (or pixels). Each grid has a value to represent its status: a point, a line, a lake, etc.

2. Vector: vector representations use the geometric Cartesian coordinate system, specifically, longitude as X and latitude as Y.

We use the vector representation in our system: GXGIS.

We apply the object-oriented technique to implement the geographical objects as shown in Fig. 1. With this technique, more specific object types are derived from the three basic types, such as country, state, city, airport, highway, etc. They are the real entities that can be found in the databases. Since the object-oriented technique supports inheritance, derived classes also have all the features defined in their parent classes including the spatial features from the classes such as points, lines and areas. Thus, these objects have both geographical/spatial information from the map and specific factual information from databases. Further, the encapsulation capability provided by the object-oriented technique allows the differences of the various objects to be hidden so that they can be manipulated in a similar manner. Thus, each object can be easily selected, retrieved, and displayed on the map.

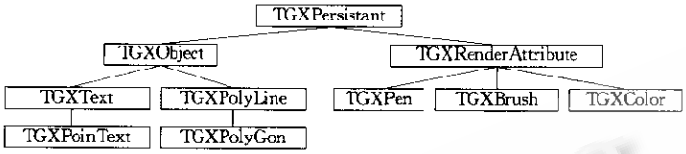

Fig. 1 Object hierarchy in GIS

This greatly facilitates the querying process.

1.2 Topology and spatial-relationship operators

Topology is the spatial relationship between connecting or adjacent coverage features (e.g., arcs, nodes, polygons, and points). For example, the topology of an arc includes its from-and-to node and its left and right polygons. An important feature of the basic geographical objects and their derived objects is that they are frequently spatially related. A city is always in a state, which in turn is in a country. A river can go through several states. Two states are adjacent to each other. These spatial relationships can be used to constrain geographical queries as well as to infer more geographic information. Using spatial relationships can significantly enhance the query capability. Some of the relationships are precisely defined and can usually be inferred or calculated from existing geographical information based on geometry theory[2]. Our system provides the following spatial operators to express these spatial relationships:

- located: point vs point.
- within: point vs line/area; line/area vs area.
- contain: area vs point/line/area.
- intersect: line/area vs line/area.
- union: area vs area.
- difference: area vs area.

These spatial operators can be used in our system to specify the spatial conditions in geographical queries. In particular, it allows the user to draw a polygon on the map and ask to 'find all the cities in this area'. Since this polygon is an arbitrary area, there may not be any predefined real entities in the databases to match it. Thus, the system has to take several steps to process this query. First, it creates a rectangular bounding box to replace the polygon, and then poses the query based on the bounding box. After answers are returned from the databases, it only returns those within the polygon (see Fig. 2). Further, with these spatial operators, geographical objects can be dynamically created when needed. For example, a query such as 'find all the cities in state X' retrieves all the cities in state X from databases, creates the city objects to record the answers and related information, and displays the objects on the map.

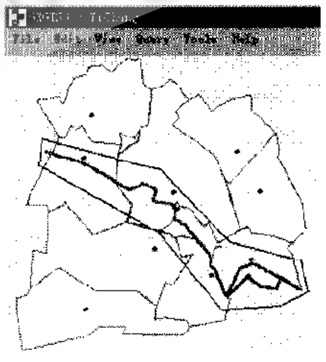

Fig. 2 Find all the cities in this area

1.3 System architecture

Figure 3 shows the overall system architecture. We will discuss the architecture in the following section.

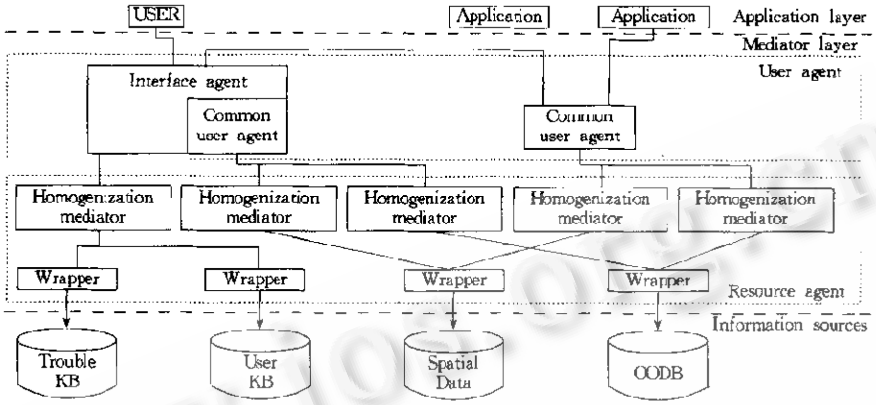

Fig. 3 Architecture of Agent-based GIS

2 Implementation of the Interface Agent

A software agent, in general, is an intelligent piece of software that carries out something on behalf of human computer users[3~5]. Here, the proposed interface agent presents users with guidance like a tutor so that even a total beginner to the GIS system can learn, during the operations, how to perform analysis of his/her interest without human support. The major functions of the interface agent include:

(1) Help users in describing the desired information in a format that is suitable to their needs.

(2) Provide approximate answers to a user's query by knowledge-based relaxation when exact answers are unavailable or high-level and imprecise queries are posed.

(3) Record the troubleshooting case data as the user experience of solving troubles, build up and show the case data to the user when a similar error occurs again[4].

2.1 Supported geographic queries

Traditionally database systems require a user to understand the database schema and only return exact answers to a user's query. Consider asking a query to a human expert of geographer. If the posed query has no answer or the complete data for an answer are not available, you do not simply get a null response. The human expert attempts to understand the gist of your query, to suggest or answer related questions, to infer an answer from data that are accessible, or to give an approximate answer. The goal of interface agent is to create information systems with these characteristics. The key is the integration of a knowledge database that represents the query semantics. More specifically, the interface agent classifies each geographic query as Fig. 4. We now present a few selected examples to illustrate the classification and the query process based on integrated view over multiple information sources (In Section 4 we will describe the integrated view).

Example 1 Question. Where is YiChang city located and what is its population size? We can directly form the SQL-like query language as the query is supported by the integrated view.

```
Select Location, population
from IntegrateViewCities.cities
where ObjectId= 'Yichang'
```

Example 2 Question. Find the population of the part of YiChang that ChangJiang River goes through? To answer the question, we have to use spatial operator 'within'. Firstly, find those places in YiChang that ChangJiang

River goes through.

```
Select ObjectId
from IntegrateViewCities.cities
where river within ObjectId
```

Secondly sumup the population of those places.

```
Select Sum(population)
from IntegrateViewCities
where ObjectId in (Select ObjectId where river within ObjectId)
```

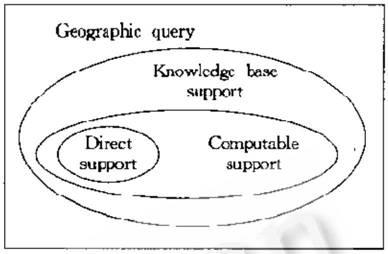

Fig. 4 Supported query

Example 3 Question. Find the population threatened by floodwater coming from ChangJiang River. To answer this question, the GIS must understand the meaning of 'threatened by floodwater'. The interface agent will match it in its knowledge database. If not matched, the agent will suggest an answer or interact with the user to acquire more information and maintain the knowledge. If the user gives more information, say, 'near the ChangJiang River and elevation is a little low', the interface agent will reason its context-sensitive knowledge-base to specify the approximated spatial operator 'near' and 'a little low'. At last, the query can be executed accurately. The whole process is illustrated in Fig. 5.

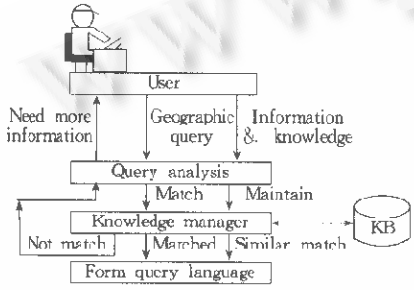

Fig. 5 Knowledge-based query

From the example shown above, the geographic query can be classified into three groups. The interface agent will process as follows.

Direct support geographic queries, these are queries syntactically supported by the integrated view.

Computable support geographic queries, these are queries that use the spatial operation as conditional constraint on the integrated view.

Knowledge-based support geographic queries, these are queries that can be executed in two steps: first a knowledge match is executed, and then a computable support or direct support geographic queries is applied to the results of the first step.

The classification of geographic queries introduces the following innovations:

• A designer can succinctly and clearly define the functionality of information sources with the three types of geographic query template as a paradigm for end-users.

• The agent automatically extends the query capability of sources that have limited functionality. This allows us to bring different sources to the same level of functionality and then it is easier to integrate them.

• The classification helps the end user to understand the integrated view.

One of the advantages of the interface agent in the system is that it can respond to a variety of user requests. Such systems allow a user to pose high-level or imprecise queries and to receive approximations when answer sets do not exist. The interface agent matches its knowledge base and interacts with user to provide domain knowledge that is associated with a given query. Yet such systems should also be able to explain how they interpreted a user's query and derived answers. In addition, a user should be able to interrogate the system to obtain a precise description of the actions it took in arriving at the answers.

2.2 Reuse of troubleshooting experience

During users' operation, the query analysis manager always takes the log of sequence of events. If the GIS reports an error, the agent will be able to detect it. When an error actually occurs, the query analysis manager knows the error code and as shown in Fig. 6, asks the troubleshooting knowledge manager to search the case base

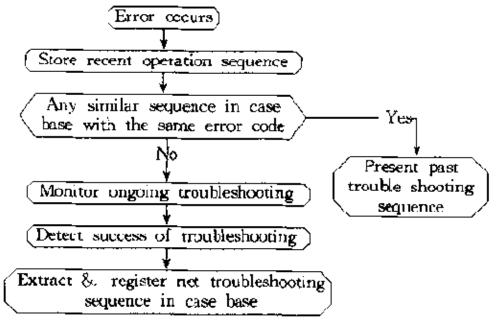
Error occurs
Store recent operation sequence
Any similar sequence in case base with the same error code
Yes
No
Present past trouble shooting sequence
Monitor ongoing troubleshooting
Detect success of troubleshooting
Extract & register net troubleshooting sequence in case base

Fig. 6 Reuse of troubleshooting experience

labeled with the same error code for a past sequence of operations that matches the current one with some tolerance. If such a sequence is found, the query analysis manager, through the progress manager, presents the user with the troubleshooting sequence of operations contained in the matched case data. If there is no such sequence in the case base, the query analysis manager keeps monitoring the ongoing troubleshooting process until the very command that caused the error is successfully executed. At that time, the agent knows that the troubleshooting has ended. Excluding the user's sequence of operations during trials and errors, it extracts the net steps taken to solve the trouble and stores them as a case data. The case data include the error code, the error message, the operation sequence that caused the error and steps taken to solve the trouble.

2.3 Architecture of the interface agent

A prototype in Fig. 7 of the proposed interface agent was implemented on a PC. The operating system is Windows NT 4.0. The knowledge manager program was written in C (partly C++), and the interface agent is implemented in Java language.

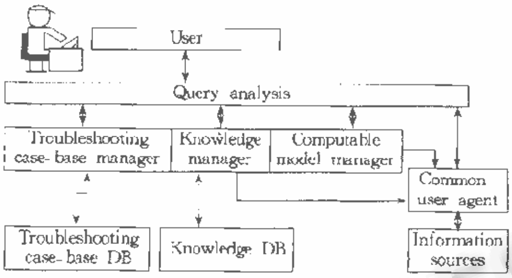

Fig. 7 Architecture of the interface agent

3 Integration with Other Information Sources

To many, the most obvious and appealing feature of GIS is the ability to present analysis results of certain domain in map form[6,7]. For certain domain there are its own information sources — data stoeed not just in standardized SQL database, but also in, e.g., object repositories, knowledge base, file system, and document retrieval system. Usually application integrated with Information Sources runs on the network infrastructures. Under a distributed computing model a few paradigms exist for facilitating integrated access to heterogeneous and autonomous data sources, notably the federated database systems (FDBs)[15], and the mediator systems[8~10].

3.1 Existing integration technologies

3.1.1 Federated database systems

Federated databases (FDBs)[15] support variations of a five-level extended schema architecture. A local schema is the conceptual schema of a component database; it is expressed in the data model of the component database. Different local schemata may be expressed in different data models. A component schema is derived by translating a local schema into a canonical data model. An export schema is a subset of a component schema that is made available to the federation. A federated schema is an integration of multiple export schemata. The external schemata are the views exposed to applications and end users. There are two types of FDBs: the tightly-coupled and the loosely-coupled.

3.1.2 Mediator systems

A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications[12]. A mediator system has the form illustrated in Fig. 8. A wrapper is a special mediator that handles idiosyncrasies of individual data sources. Today, many mediator systems are being built[8~10]. Unlike federated databases that often aim at creating a single database illusion, mediator systems offer specific information services. Such services can be built based on existing services provided by other mediators. Thus, the mediator system is not monolithic but is a network of mediators, each accessing multiple heterogeneous data sources and/or other mediators. Compared with FDBs, the mediator systems put more emphasis on flexibility, versatility, and knowledge-based capabilities. Domain specific mediators (the so-called vertical mediators) often exploit domain knowledge to enhance usability and performance.

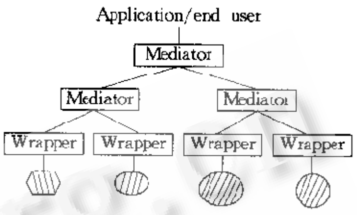

Fig. 8 Mediator systems

3.2 GIS using 2-tier mediation system

When integrating information sources in GIS, we distinguish between two categories of heterogeneity among data sources[13]: schematic mismatches that arise when the same application domain is modeled differently; and instance level conflicts that arise when inconsistent values for the same real world object are recorded. Schematic mismatches must be resolved before instance level conflicts are tackled. The process of resolving schematic mismatches is referred to as homogenization. In GIS, specialized mediators, the homogenization mediators, support this process. The result of homogenizing a source is a homogenizing view that hides the deviations of this source from the target application view in both structure and semantics. The integration mediator 'glues' a large number of data sources together by combining all the homogenizing views into an integrated view. Since all the sources are homogenized, the integration process is greatly simplified; it only considers instance level conflicts; and schematic heterogeneity has been largely removed by homogenization as shown in Fig. 9.

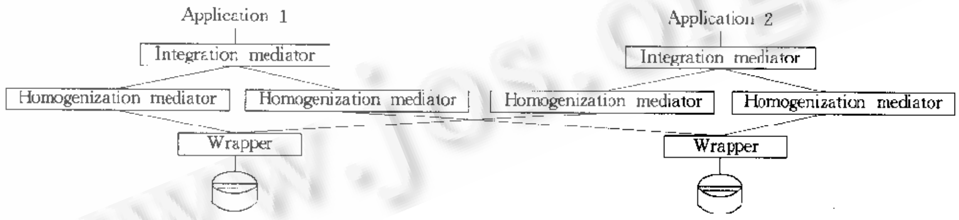

Fig. 9 2-tier mediation system

4 Agent-Based Architecture

4.1 Overview of the agent-based system

We have developed a prototype system, called GXGIS, using Agent-based technology as shown in Fig. 10. The GXGIS application consists of a collection of agents, aiming at portability and compatibility with popular Web Browser. The Agents can be organized as the user agents (those appearing to the left of the cloud) that actas proxies for individual users or specific domain applications, the resource agents (those appearing to the right of the cloud) that serve as interface to information sources and broker agents (those enclosed by the cloud).

4.2 Resource agent and homogenizing view

The resource agents resolve schematic mismatches and provide homogenizing view[13]. The process of resolv-

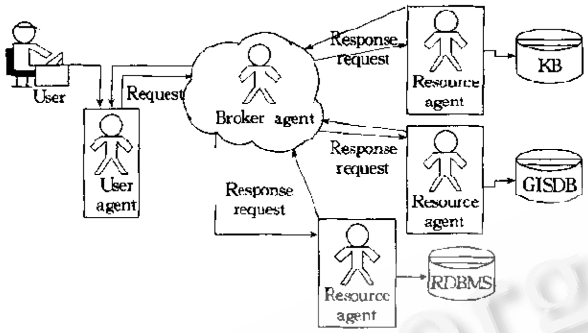

Fig. 10 Agent-Based architecture

ing schematic mismatches is referred to as homogenization. The result of homogenizing a source is a homogenizing view that hides the deviations of this source from the target application view in both structure and semantics. An example is given about the homogenizing view. Consider a spatial database storing the geographical information of cities and the relation DBMS manages the information of cities on population and Gross Domestic Product (GDP) annually. The local schema can be described as follows:

```
Class city_spatial
  type tuple(
    objectid:string;
    ObjectData:Pointlist;
    Objecttype:string;
    RenderAttribution:tuple(pen:TGXPen;
    Brush: TGXBrush; color:TGXColor)
    )
            SpatialDB schema
```

```
Relation city_info(Name:string;
      Population:longint;
      )
Relation city_GDP(Name:string;
      Year:integer;
      GDP:float;
      )
            RelationDB schema
```

We can give the homogenizing view with the specification of extensible view[14] for the two data sources as follows:

```
define LocalInterface LVcity_spatial as{
structure    {type city_spatial: tuple(objectid:string; Objecttype:string; ObjectData:Pointlist;
              RenderAttribution:tuple(pen:TGXPen; Brush: TGXBrush: color:TGXColor)) }
Access       {type city_spatial⇔type city_spatial in SpatialDb
              type ObjectData⇔type ObjectData in SpatialDb          }
Semamtic {    objectid in type city_spatial⇔name of type city_info in LVcity_info
              objectid in type city_spatial⇔name of type city_GDPset in LVcity_info } }
define LocalInterface LVcity_info as{
structure {   type city_info: tuple(Name:string; Population:longint;)
              type city_GDPset: set (city_GDP)
              type city_GDP: tuple(Name:string;Year:integer;GDP:float;)        }
Access {      type city_info⇔type city_info in RDBMS
              type city_GDPset⇔type city_GDPset in RDBMS          }
Semamtic{     name in type city_info⇔objectid of type city_spatial in LVcity_spatial;
              name in type city_GDPset⇔objectid of type city_spatial in LVcity_spatial;}}
```

4.3 Common user agent and integrated view

The common user agents are responsible for integrating a large number of data sources together by combining all the homogenizing views into an integrated view[13]. Since all the sources are homogenized, the integration process is greatly simplified; it only considers instance level conflicts; and schematic heterogeneity has been largely removed by homogenization. Building a common user agent means building an integrated view. The common user agent also includes a query processor, which converts geographic query based on the integrated view into homogenizing views. The example followed shows the integrated view combined from the two homogenizing views above.

```
define IntegratedView IntegrateViewCities as{
structure {     type cities ;tuple (objectid:string; location:pointlist;
                population:longint; city_GDPset; set (city_GDP) ) }
Semamtic        {objectid of type cities⇔
                objectid of type city_spatial
                in LocalInterface LVcity_spatial;
                location of type cities⇔
                location of type city_spatial
                in LocalInterface LVcity_spatial;
                population of type cities⇔
                population of type city_info
                in LocalInterface LVcity_info;
                city_GDPset of type cities⇔
                city_GDPset
                in LocalInterface LVcity_info;}
}
```

4.4 Broker agent

The broker works as distributed components that communicate and cooperate via Object Request Broker (ORB), as shown in Fig. 11. One of the primary jobs of a broker is to maintain a repository containing current and correct information about operational agents and the services they can provide. When an agent comes online, it announces itself to broker by advertising to it. The broker stores all of the advertised information in its repository. Figure 12 shows some of the information that can be advertised by a broker agent. When an agent's set of available service changes, the agent may update its advertisement, and the broker will update the information in its repository. When an agent goes offline, it will first unregistered itself from the broker. Also, the broker periodically pings each of the agents that have advertised to it, to discover any agent that has failed. The broker removes from its repository all information about agents that have failed or unregistered themselves. The broker will make the GIS more scalable and efficient.

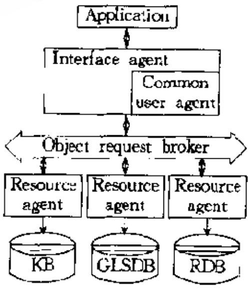

Fig. 11　Broker in GIS

Agent name and location
The unique identifier for the agent
Directions on how to contact the agent (host, port, transport protocol)
Agent type (e.g., broker agent, resource agent, user agent)
Agent syntactic knowledge
Communication languages/services (e.g., KQML, CORBA)
Content languages (e.g., SQL, geographic queries)
Agent capabilities
The conversation types it can participate in (e.g., ask-all, subscribe, tell)
The capabilities of the agent (e.g., Direct support, Computable support, Knowledge-based support)
Agent content
Supported ontologies (e.g., economic)
Agent properties
Adaptivity (e.g., cloneable, mobile, able to operate disconnectedly)

Fig. 12 Advertised information

5 Conclusion

We have developed a prototype system[16]——GXGIS, using distributed agent architecture. Our implementation so far has served two purposes:

(1) The interface agent processes three types of geographic queries. The classification helps to define the functionality of information sources with the three types of geographic query template as a paradigm for end-users. The agent matches its knowledge base and interacts with user to provide domain knowledge that is associated with a given query. Also the agent records user's troubleshooting experience and show it to the same users and other users as hints.

(2) The broker-agent architecture of GIS can meet the needs for integration of heterogeneous information sources. The agents can be organized as user agents, resource agents and broker agents. Also we model mediation as a 2-step process: homogenization followed by integration, each performed by respective mediation. The broker and the 2-tier mediation model make the GIS more scalable and efficient.

Acknowledgements The research and development of GXGIS is supported by YiChang City Government. YiChang is a large city and has a population of 4 million. The famous Three-Gorge Project is located in YiChang city.

References:

[1] Bennett, D. A. A framework for the integration of geographic information system and modelbase management. International Journal of GIS, 1997,11(2):143～153.

[2] Wesley, W., Chu, Hua-yang, Kuorong, Chiang, *et al*. CoGIS: a cooperative geographical information system. In: Proceedings of the SPIE Conference on Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry. Orlando, FL: SPIE Press, 1994. 1413～1425.

[3] Zhang, Guo-gen, Chu, Wesley W., Frank, Meng, *et al*. Query formulation from high-level concepts for relational databases. In: IEEE Computer Society; Proceedings of the User Interfaces to Data Intensive Systems (UIDIS). Edinburgh, Scotland, 1999.

[4] Miyoshi, A., Yagawa, G. An interface agent that actively supports CAE beginner users in performing analysis. Advances in Engineering Software, 1999,(30):575～579.

[5] Lieberman, H. Integrating user interface agents with conventional application. ACM Conference Intelligent User Interfaces, San Francisco, 1998,(1):6～9.

[6] Miles, Scott. B. Application and issue of GIS as tool for Civil Engineering modeling. Journal of Computing in Civil Engineering, 1999,(7):144～152.

[7] David, A. Holdstovk. Basic of Geographic Information System. Journal of Computing in Civil Engineering, 1998,(1):1～5.

[8] Hammer, J., Breunig, M., Garcia-Molina, H., *et al.* Template-based wrappers in the TSIMMIS system. In: Joan Peckham, ed. Proceedings of the 26th SIGMOD International Conference on Management of Data. Tucson, Arizona: ACM Press, 1997. 12～15.

[9] Garcia-Molina, H., Hammer, J., Ireland, K., *et al.* Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In: Proceedings of the AAAI Symposium on Information Gathering. Stanford, California: AAAI Press, 1995. 61～64.

[10] Chawathe, S., Garcia-Molina, H., Hammer, J., *et al.* The TSIMMIS project: integration of heterogeneous information sources. In: Proceedings of the IPSJ Conference. Tokyo, Japan, 1994,(10):7～18.

[11] Papakonstantinou, Y., Gupta, A., Garcia-Molina, H., *et al.* A query translation scheme for rapid implementation of wrappers. In: International Conference on Deductive and Object-Oriented Databases. 1995.

[12] Quass, D., Rajaraman, A., Sagiv, Y., *et al.* Querying semistructured heterogeneous information. In: International Conference on Deductive and Object-Oriented Databases. Singapore, IT-Asia, 1995.

[13] Wesley W. Chu, Hua, Yang, Kuorong, Chiang, *et al.* CoBase: a scalable and extensible cooperative information system. Journal of Intelligence Information Systems, 1996,(6):223～259.

[14] Yen, Cheng-huang, Miller, L.L. An extensible view system for multidatabase integration and interoperation. Integrated Computer Aided Engineering, 1995(2):97～123.

[15] Temlleton, M., Henley, H. InterViso: dealing with the complexity of federated database system access VLDB. In: Proceedings of the 21st International Conference on VLDB. Switzerland, Morgan, Kaufmann, 1995,4(2):287～317.

[16] Feng, S., Xu, L.D. An intelligent decision support system for fuzzy comprehensive evaluation of urban development. Expert System with Application, 1999,16(1):21～32.

一种基于 Agent 结构的地理信息系统

唐 超[1]，冯 珊[1]，许立达[2]

[1](华中科技大学 控制科学与工程系 系统工程研究所，湖北 武汉 430074)

[2](*Department of Management Science and Information System, Wright State University, Dayton, USA*)

摘要：把智能体(Agent)技术引入地理信息系统中，用以增强地理信息系统的人机交互能力和多信息源的可扩展性.用户的问题与用户的查询目标和领域知识相关，因此，交互界面智能体能通过匹配知识库的过程引导用户细化问题，最终提供给用户满意的答案.而且，基于智能体体系结构的建立能够更好地集成各种异构信息源，由于各种异构信息源的集成，所以可以提供给用户数量更多和质量更高的信息.此外还详细描述了一个基于智能体结构的原形系统 GXGIS.

关键词：界面智能体；地理信息系统；知识库；异构信息源；调解器

中图法分类号：TP391　　**文献标识码**：A