

# Geo-Agents: Design and Implement

Ding Li, Qu Lei, Zhang Ying, Luo Ying-wei, Wang Xiao-lin, Xu Zhuo-qun  
Artificial Intelligence Lab in Department of Computer Science & Technology,  
Peking University, Beijing, 100871, P.R.China

**Abstract:** In this paper, Geo-Agents, a multi-agent system that processes distributed geospatial information and geospatial service, was presented. Firstly, the requirement for distributed geographical information process was discussed, and the architecture of Geo-Agents was introduced. Then, in-depth discussions were raised on agent system implementation, such as the basic agent, agent advertising, message passing, and collaborating. An example was also given to explain the problem solving process.

**Keywords:** Geographical Information System (GIS); Multi-Agent System (MAS); Geo-Agents; Communication; Collaboration

## 1. Introduction

The concept of agent is becoming increasingly important not only in research (where it has been in use for some time) but also now in commercial applications.

There exist several definitions of agent technology. To present here we chosen Jennings' <sup>[1]</sup> because although it is a very general definition it includes key features that will provide an initial classification of agent technology.

*An agent is a self-contained problem solving entity implemented in hardware, software or a mixture of the two that should include the following characters:*

- *Autonomy: agents should be able to execute their problem solving tasks without direct intervention from humans or other agents. They should, to some degree, control their own actions and internal state.*
- *Social ability: agents should be able to interact, when they see fit, with other agents and humans, either to complete their problem solving or to help other agents.*
- *Responsiveness: agents should hold knowledge on their environment and be able to respond to any changes that may happen in this environment.*
- *Proactiveness: agents should not only act in response to their environment, they should be able to take advantage of fortuitous opportunities to achieve their designated goals. An agent should be able to modify its behavior in response to stimuli.*

Just because of possessing these characters, using agent technology in Distributed Geographical Information System (DGIS) appears natural. Several problems have arisen in the development of DGIS where the creation and use of intelligent agents may be successful <sup>[2]</sup>.

- Location and retrieval of Spatial Information in large networks (and specifically the Internet). With the intense spatial information, every operation or analysis of spatial data may involve large volumes of distributed data. It makes difficult for the process of the data management. Different agents can be built to locate the necessary information for the user and filter the retrieved information by identifying the users necessities
- Improve the cooperation ability in Geographical Information System (GIS). With the extension of GIS application, it is impossible to finish one task by single function. It will be easier to finish some tasks by several agents' cooperation.
- Acting in a personalized manner. The instructions they receive from their specific user are stored to be compiled later on and repeated when necessary. In this way, agents can execute new tasks using the user's preferences.

As a proof of concept, we design and implement a prototype agent-based DGIS environment Geo-Agents. Section 2 introduces the Geo-Agents' architecture; Section 3 discusses implementation issues, including basic organization, advertising, messaging and collaboration; Section 4 gives an example about Overlay operation in GIS and Section 5 lists some related works.

## **2. Geo-Agents Architecture**

Geo-Agents is a Multi-Agent System. It is composed of a group of software agents, who are capable of collaboratively completing a task in GIS domain. These agents may all be running on the same physical machine, or may be distributed across a network. So Geo-Agents also has the characteristic features of Distributed System.

In order to complete a specified task, agents in Geo-Agents should collaborate with each other. Some agents may talk with outside actors, such as human users, agents in other agent systems and data sources. Inside the Geo-Agents boundary, agents play different roles, such as the manager, the coordinator, the executor, and etc. Fig 1 outlines the generic architecture of Geo-Agents, in which agents are classified due to their abilities and roles.

There are five types of agents who fall into two categories: administrative agents and executive agents. Administrative agents include Administrator and Facilitator. They keep alive all the time. Their responsibilities are balancing system load, facilitating communication, managing executive agents, and etc. Executive agents include UI

agent, task agent, and domain agent. They are temporary agent. Their responsibilities are solving an ad hoc problem, e.g. searching data, and etc.

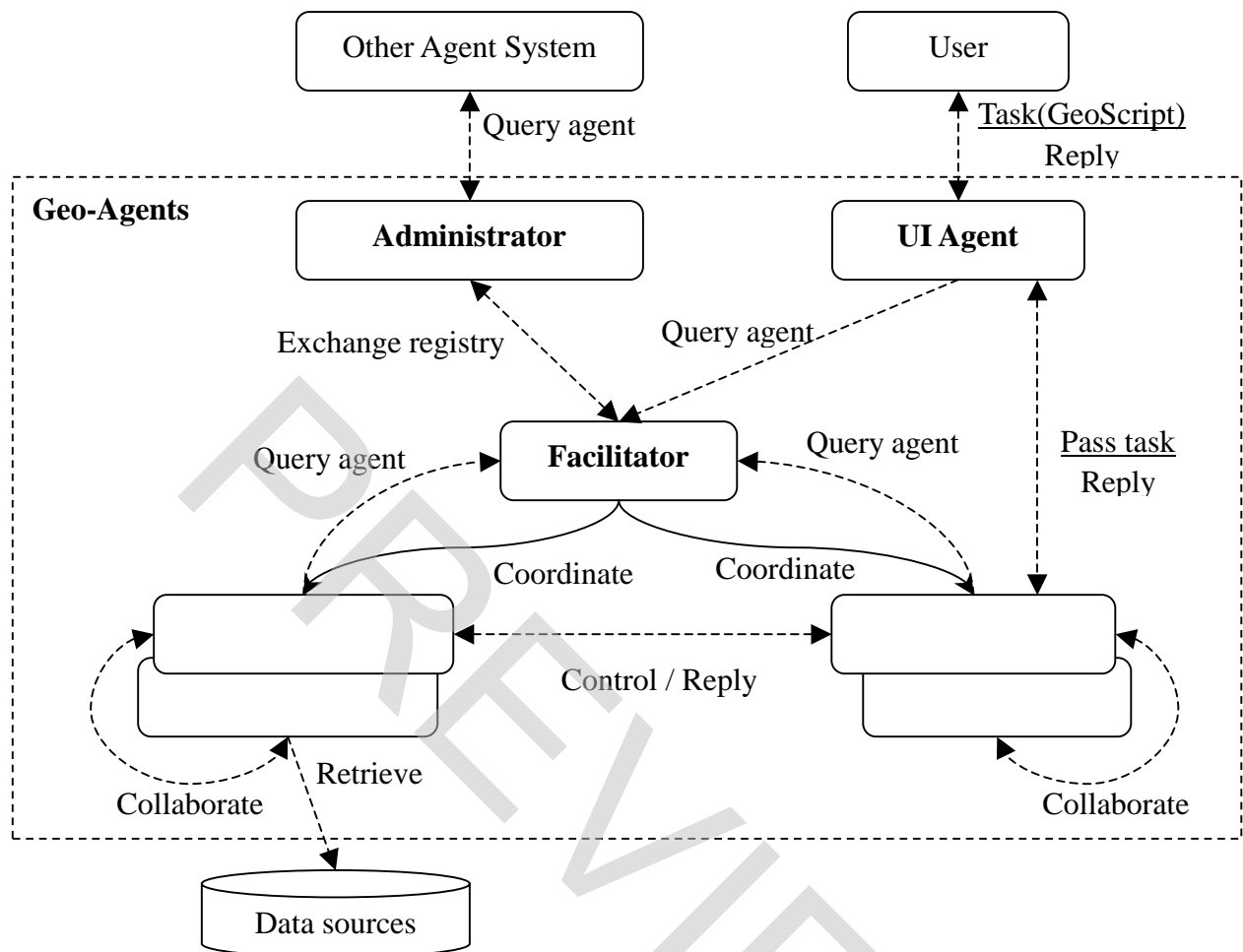


Fig 1: Geo-Agents Architecture

- (1) **Facilitator** — Each host in Geo-Agents possesses a Facilitator. As an administrative agent, it is designed to deal with host affairs. All *query agent* messages should be sent to a Facilitator, and then all Facilitators in Geo-Agents collaborate to look up or create the specified agent's instance, and then return a reference to the requester. While creating a new agent, an algorithm is used to balance the load among all the hosts.
- (2) **UI Agent** — Geo-Agents can communicate with human users via User Interface Agent (UI agent). A simple script language, GeoScript<sup>[3]</sup>, was introduced to help users to describe GIS tasks. GeoScript can not only depict the task with facts, rules, and goals, but also can instruct domain agents' collaboration, and depict the relationships among Task agents. So if a user wants the help of Geo-Agents, he or she may write task description in GeoScript format, pass it to a UI agent. UI agent will do everything, and what left the user to do is simply waiting for the result.
- (3) **Task Agent** — A Task agents is used to complete a task. It can obtain a task from a user via UI agents or the other task agent. Task requester should query task agent

first, then send task to the task agent. When received a task, task agent interprets and executes the script, collaborate with other task agent, schedule domain agents to collaboratively complete the task, and return the result to the requester.

- (4) Domain Agent — Domain agents are designed to provide domain specific services. Usually, a domain agent in Geo-Agents can only provide a well-known, predefined service in GIS, such as overlay of map layers, searching shortest path between two sites in a map, connectivity analysis, and etc.
- (5) Administrator — There is only one administrator in Geo-Agents. It is also an administrative agent, which is designed to deal with sophisticate communication affairs. It has two responsibilities. One is to serve as an inter-operator. It provides the communication interfaces to outside agent systems. All queries from or to outside agent systems are performed via its services. The other is to act as a census provider. It gathers and records the information from all facilitators in the whole agent system, such as agent grouping information, agent registration information, and agent instances distribution information. Then facilitators can use these data as the basis of creating new agent.

### **3. Implementation**

Agents in Geo-Agents are functionally different because they play diverse roles. However, they possess something in common. To be qualified as an agent, each one must have the basic abilities of advertising, messaging. So Geo-Agents needs a basic agent (BA), and every agent should derive from it and inherit its abilities.

#### **3.1 Basic Agent**

A BA should cover the minimum requirements of an agent. It should advertise itself, communicate with other agents, and do something to achieve its goal. Multi-thread mechanism and DCOM technologies are used to implement BA. They not only provide an extensible and flexible organization for BA, and improve the performance of BA. Fig 2 shows the organization of a BA.

- (1) Receive Message Thread (RMT) – It is a thread that listens and receives messages from other agents.
- (2) Receive Message Queue (RMQ) – It is a queue that buffers incoming messages.
- (3) Send Message Thread (SMT) – It is a thread that sends messages to other agents.
- (4) Send Message Queue (SMQ) – It is a queue that buffers outgoing messages.
- (5) Process Message Thread (PMT) – It is a thread that handles an incoming message, usually do something achieve a goal.
- (6) Thread Manager (TM) – It manages all threads in an agent.
- (7) Wait Thread Manager (WTM) – It manages paused thread waiting for

synchronizing messages.

(8) Sub Agent Manager (SAM) – It manages the sub agents of this agent.

(9) Components – They provide GIS or other functions and do the real job.

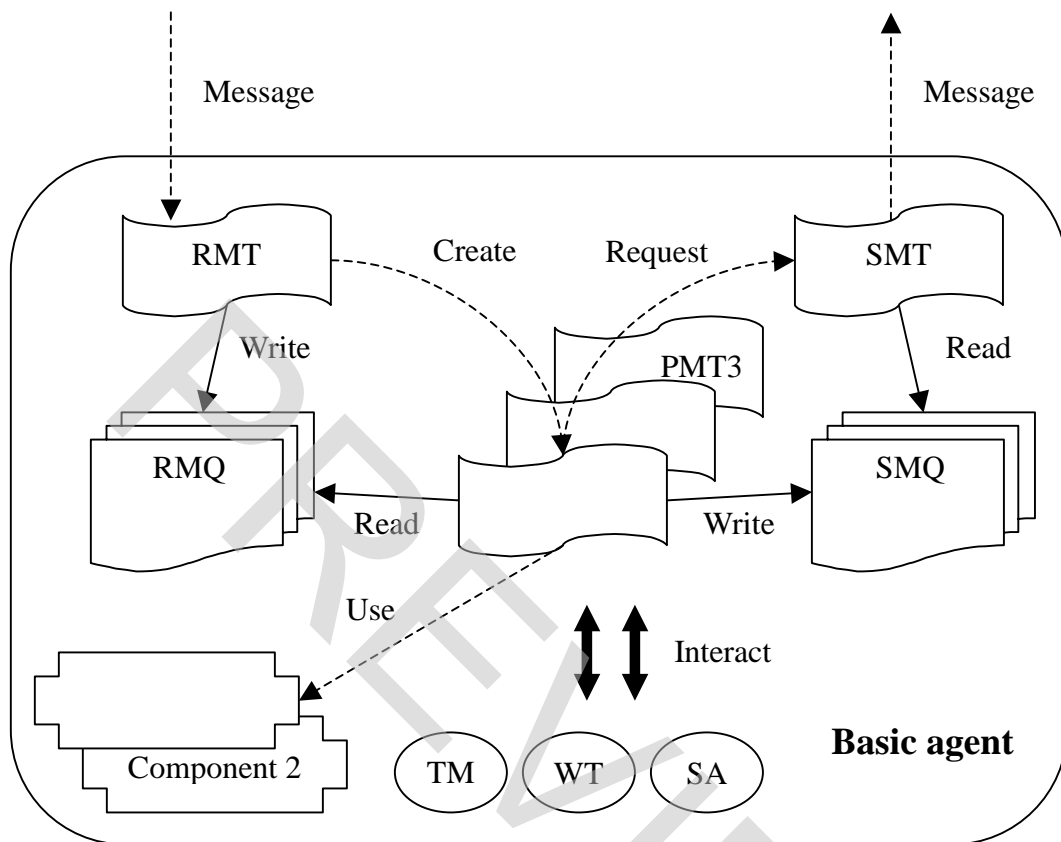


Fig 2: The organization of a basic agent

The BA uses message to communicate with others. The essential process of BA is message loop: receiving messages, processing messages, and sending messages.

An incoming message is received by RMT. Then RMT puts the message into RMQ and starts a PMT to process the message.

The PMT reads the message from RMQ. Following the instruction of the message, the PMT uses some components to complete the task. When the PMT finishes its job, it writes the result to SMQ and requests the SMT to send. To add new abilities to an agent, we can simply add some components and add new “case“ branches in its PMT class.

Finally SMT sends the result as an outgoing message.

Since a Message Queue is a shared resource, it may be access by several threads simultaneously. A Read/Write lock is used to provide exclusive access to it and ensure

the sequence of messages. In this case, R/W lock is implemented by P/V operation on a signal variable.

## **3.2 Key Issues**

### **3.2.1 Advertising**

Advertising is the essential feature of an agent. The agents in Geo-Agents advertise themselves in two ways: static and dynamic.

As mentioned in section 2, administrative agents advertise themselves in static way, so they have already known each other's address. That is to say, an Administrator has a list of all Facilitators' addresses. With the list, the Administrator can connect and communicate with all Facilitators.

On the other hand, executive agents can advertise themselves in both ways. Static advertised executive agents provide the basic service of Geo-Agents. If the user like, he or she can make a personalized agent and let the agent join Geo-Agents via dynamic advertising.

The process of executive agent advertising is a bit complex. They should advertise themselves on one or several Facilitators. The Administrator gathers all executive agents' advertisement, and maintains two tables: agent registry table (ART) and agent distribution table (ADT). ART records advertising information of all executive agents, such as their abilities, the host supports them, the preferred host and the alternative host where they should be created, and etc. ADT records the statistical information about all executive agent instances, such as the memory they occupied, the age, and etc.

Moreover, since Geo-Agents has an extensible architecture, new executive agents can dynamically join a Geo-Agents in a convenient way.

### **3.2.2 Communication**

As mentioned in BA, Geo-Agents use message mechanism to communicate. A six-item format  $\langle From, To, msgType, replyWith, replyTo, msgContent \rangle$  is used to describe a message. *From* is the sender's ID. *To* is the receiver's ID. *msgType* indicates the type of the message. *ReplyWith* and *ReplyTo* are used to support the collaboration mechanism. They are optional items and will be discussed in fig 4. *msgContent* records the content of message, it was written in a format which can be understood by both the sender and the receiver. Usually, such format depends on the sender, the receiver and the message Type.

According to the needs of GIS and agents, the messages in Geo-Agents can be divided into four categories:

- (1) Management messages – They are used to manage executive agents in Geo-Agents. The sender or the receiver or both must be administrative agent.

For example, CREATE\_AGENT, AGENT\_START, STOP\_AGENT, TASK\_END, and etc.

- (2) Collaboration Messages – They are used to support the collaboration among task agents. For example, FIND\_TASKAGENT, IS\_GROUP\_MEMBER, SENDTO\_GROUP, COOPERATE\_END, RET\_MEMBER\_NUM, and etc.
- (3) UI Messages – They are used to pass GeoScript among UI agents, Facilitator, and task agent. For example, EXECUTE\_SCRIPT, GENAGENT\_START.
- (4) Domain Messages – They are used to support the interaction among domain agents. For example, AGENT\_PARAMETER, AGENT\_RULE, and etc.

Geo-Agents use synchronization point (Sync Point) to support the collaboration mechanism. When an agent needs the result of another agent to continue its job, it must be paused and wait for reply message from the specified agent, and this action is called entering the *Sync Point*. Fig 3 shows the state transition of entering and leaving the *Sync Point*.

When entering the *Sync Point*, a unique ID is generated to identify the *Sync Point*. Then a *Query* message is sent to the specified agent, and its *ReplyWith*'s is filled with *Sync Point* ID. The working thread is paused and transferred from the TM to the WTM. And then when RMT receives a message, it compares the value of *ReplyTo* with the *Sync Point* ID. If they are equal, the paused thread is waked up and continues its previous job.

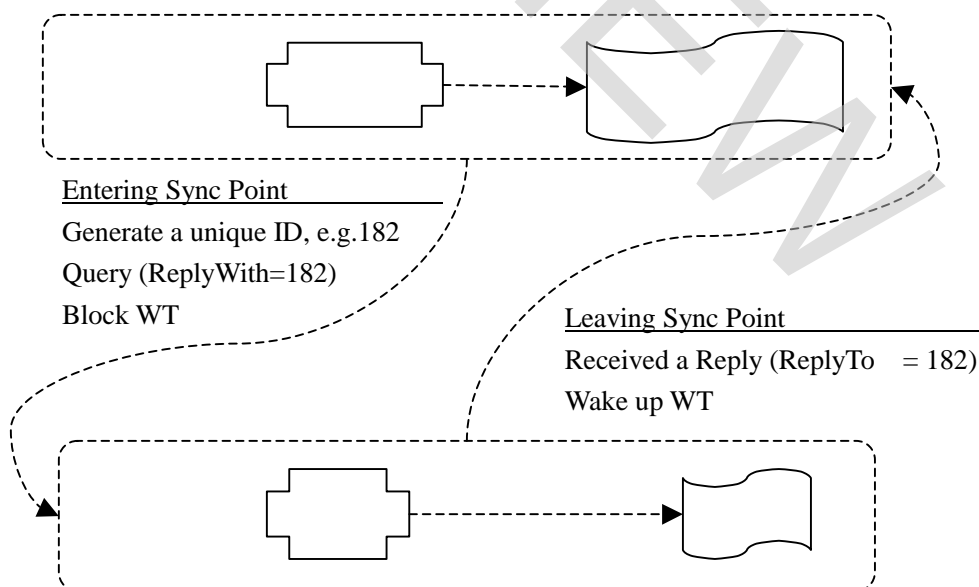


Fig 3: Entering and leaving *Sync Point* (state diagram)

### 3.2.3 Create an agent instance

Considering the complex structure of the spatial data and the space and time cost of Geo-processing, new agent should create on the least loaded host, thus to achieve the best performance of the whole agent system. Facilitators and Administrator collaboratively provide an algorithm to find a proper host so as to balance system load. Fig 4 shows how an executive agent's instance was created.

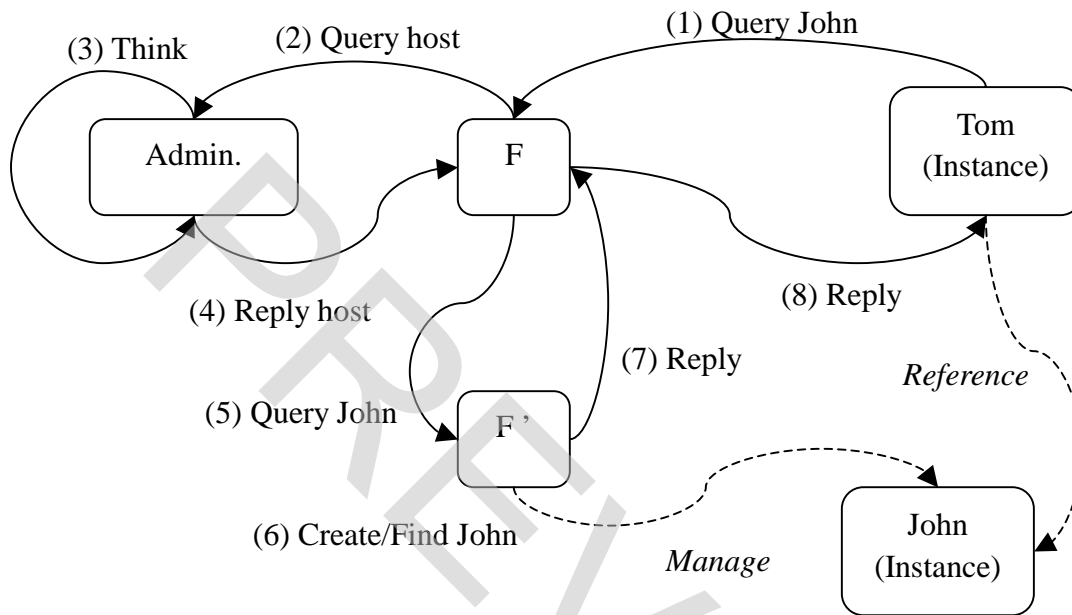


Fig 4: The process of creating an agent instance

*Admin.* – an Administrator

*F* – a Facilitator

*Tom* – an agent

*F'* – another Facilitator

*John* – another agent

- (1) Tom => F: [Query] I need a Collaborative Agent (John).
- (2) F => Admin: [Query] Please tell me where can I get John.
- (3) Admin: [Think] Search in agent registry table:
  - a) Does John have a preferred host, if yes then host is found
  - b) If a) answers no, does John have an alternative host, if yes then host is found
  - c) If b) answers no, list all hosts that support John, and select the minimum used one.
  - d) If c) fails (can not find any host), then search failed.
- (4) Admin => F: [Reply] Ok, here is result (success with a host/ fail)
- (5) F => F': [Query] Please create an instance of John for me
- (6) F': [Action] Create John
- (7) F' => F: [Reply] here is John's reference.
- (8) F => Tom: [Reply] here is John's reference.

### 3.2.4 Collaboration

Collaboration is an important feature of Geo-Agents.

Task agent can collaborate. They can collaborate in three ways: peer-to-peer (fig5.a), group (fig5.b), centralized group (fig5.c). Administrator helps group collaboration. It has an agent group table, which records the members (Task agents) in a group. It also provides some group management services, such as querying group member, group broadcast, querying group size, joining a group, and etc.

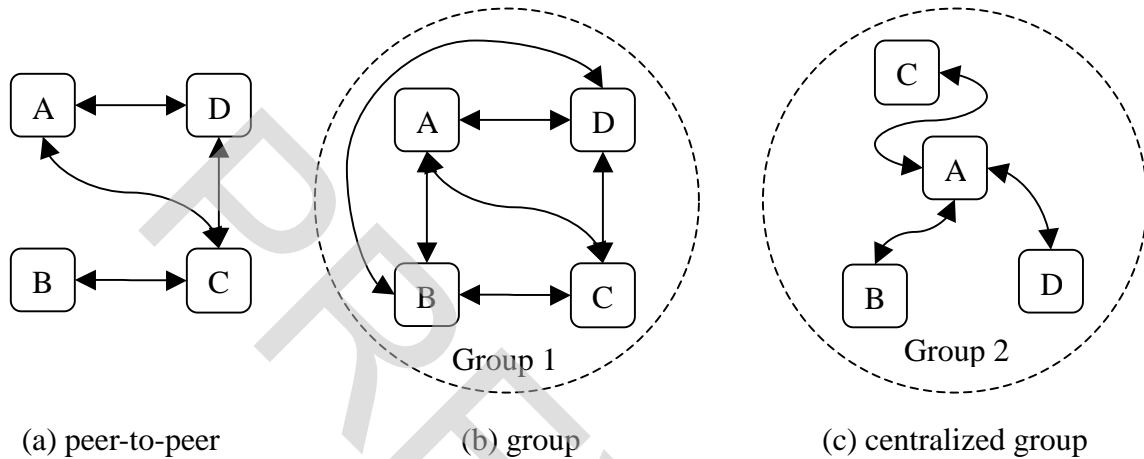


Fig 5: Collaborations among Task agents

Domain agents collaborate under the control of task agent. They use hierarchical model, which has a manager and some workers (or sub-agents). Sub-agents' life cycles are within their manager's life cycle. A domain manager agent (DMA) can create sub-agent via Facilitator. If the DMA wants to kill his sub-agents or the DMA is dying, it can send STOP\_AGENT message to Facilitators to kill his sub-agents.

When a task agent meets a big problem, it can divide the problem and schedule domain agents to solve the sub-problems. When receive a problem, domain manager agent may use one or more worker agents to get solution. Fig 6 gives a typical view on how Task agents and Domain Agents organized to collaborate.

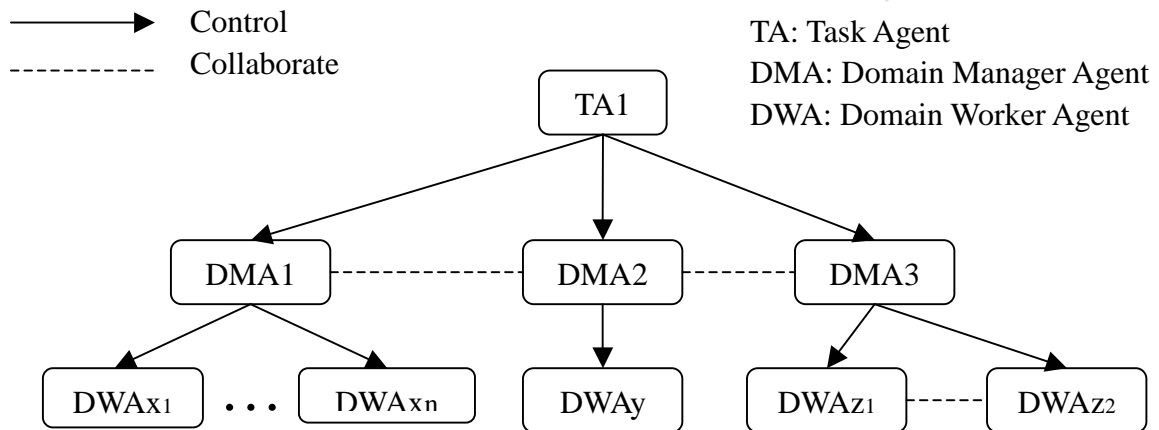


Fig 6: Collaborations among domain agents controlled by a task agent

## 4. An Example

Geo-Agents is designed to solve GIS problems efficiently. Here is an analysis on solution of a typical GIS problem — Overlay operation. Overlay operation overlays two or more map-layers following some particular rules. In the traditional way, it is a time consuming task because the operation is much complexity and need to transmit all of the map-layers to the local machine, and then overlay the map-layers one by one.

We can complete the overlay task more efficiently in Geo-Agents. Two kinds of Domain Agents can be designed to perform this task - overlay manager agent (OMA) and the overlay agent (OA). Usually, OMA needs the help of data searching agent (DSA) to locate and access the map-layer the task needs. Their functions and relationships can be described as following:

- OMA is responsible to interpret GeoScript, and then manipulate the OA and DSA to complete the task. The policy of dispatching OA will affect the system performance obviously. OMA should try its best to reduce the transmission of map-layers, and make use of all of the hosts' resource available as sufficiently as possible. OMA will return the result map-layer or an error message to the user finally.
- OA overlays the two map-layers assigned by the OMA, saves the result map-layer as a temporary layer in the spatial database and then returns the result map-layer link to the OMA.
- In order to help an OMA to access the map-layers it operates, several DSAs may be used to search in spatial metadata database and return locations of the map-layers to the OMA.

Fig 7 is the collaboration diagram of OMA, OA and DSA, which describes the process of overlay in detail.

- (1) User describes an overlay task in GeoScript format, which includes map-layer links and rules.
- (2) While interpreting GeoScript, OMA creates an empty map-layer link pool. The pool contains links of accessible map-layers. OA can access these map-layers via the links.
- (3) If OMA finds any map-layer not accessible, it creates a DSA to search.
- (4) The DSA looks up map-layer's address in the spatial metadata database.
- (5) The DSA puts the map-layer link to the map-layer link pool.
- (6) Once there is more than one links in the map-layer link pool, OMA starts an overlay operation. With the policy of minimizing data transmission, OMA divides

map-layers into several pairs, chooses a host for each pair. Then OMA creates an OA on each host to overlay the pair. All OAs can run independently and concurrently. When there is only one link in the map-layer link pool, and all DFAs, OAs have finished their jobs, the overlay task is successfully finished.

- (7) An OA gets two map-layers from the spatial database (the map-layer might reside on another machine because of the distribution of the map-layers), and then overlays them. The result is a new map-layer.
- (8) The OA saves the result as a temporary map-layer to local spatial database.
- (9) The OA puts the link of the result map-layer to the pool, and then kills itself.
- (10) If overlay task is completed successfully, OMA returns the result map-layer to the user. Otherwise, the user will receive an error message.

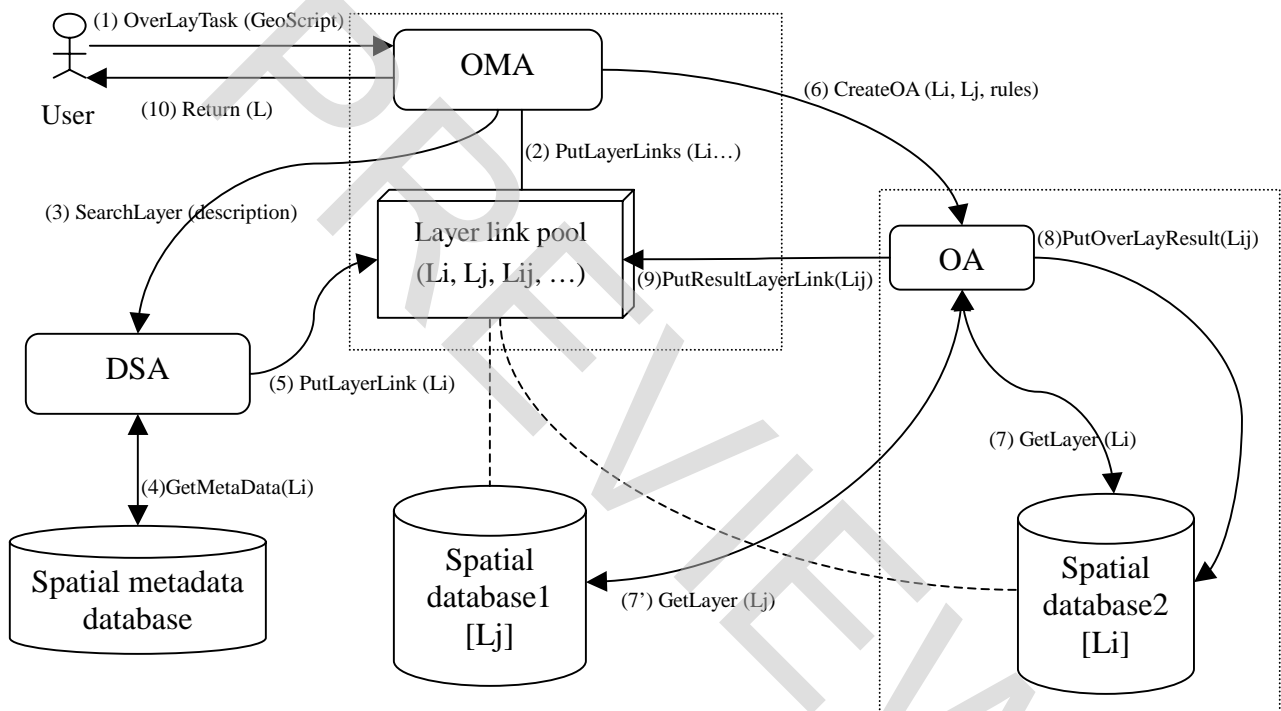


Fig 7: Collaboration diagram of OMA, OA and DSA

This example shows the advantages of Geo-Agents in solving GIS problem. Creating OAs on the hosts where data located may dramatically reduce data transmission. Running OAs on different hosts can make maximum use of all hosts in the system, thus improving the performance of the system.

## 5. Related works

Many research works have been done for the implementation and design of agent technology. The followings are two works related to DGIS.

The MEGAAOT (Modeling Geographic Elements for Environmental Analysis in Land use Management) project <sup>[4]</sup> was identified as a good example for an experience

in Multi-Agents Systems because it integrates the three relevant areas of research: Simulation of changes in the characteristics of elements (resulting in the generation of relevant information for making decisions in environmental planning) and the creation of dynamically managed applications that will improve the interface provided by the chosen GIS package. Initially it was conceived to applied at the municipal level and implemented on GIS generic platforms, defines intelligent connections between a GIS database, analytical operators and environmental models to create environmental interrelated scenarios. The project was implemented as a spatial decision support system for environmental and municipal planning focusing on the possibility of simulating the effects of human actions and land use transformations on an interactive basis.

Sugumaran<sup>[5]</sup> presented a conceptual model for a distributed intelligent agent-based SDSS, and are currently implementing a prototype environment to demonstrate the feasibility of such a system.

## **6. Conclusion**

Agents are already being used in key areas of software development. Everywhere agents are developed for Information retrieval, Intelligent networking, Execution of tasks, Collaborative processing, etc.

This paper presented Geo-Agents, which is an implemented multi-agent system specialized in geospatial information processing. It focuses on the architecture and implement works of Geo-Agents. The implementation of Geo-Agents demonstrates the feasibility of such a system, and the running result of Geo-Agents is also satisfied. The Geo-Agents is an easy-use, scalable, extensible DGIS.

Work in this area is only beginning. There are still many problems that need to be addressed, such as security problem, balancing system load and so on. But it is our belief that the use of agent technology in DGIS has enhanced its performance and ability. The agent technology is now an efficient approach to improve DGIS.

## **7. References**

- [1] Jennings, N.R., Agent Software, Proceedings of The Agent Software Seminar, London, England, UNICOM Seminars, 25-26 April, 1995.
- [2] Rodrigues, A., Raper, J., Capit, M., Implementing Intelligent Agents for Spatial Information, Proceedings of The Joint European Conference on Geographical Information, The Hague, The Netherlands, 26-31 March, 1995.
- [3] Luo Ying-wei, A Study on Agent-based Distributed GIS,[Ph. D. Thesis], Beijing,

Peking University, 1999 (Ch).

[4] Armanda Rodrigues, Cric Grueau, Jonathan RAPER, Nuno Neves, Environmental Planning using Spatial Agents, Department of Geography, Birkbeck College 7/15 Gresse St., London W1P 2LL, England, 1998. <http://www.dh.lnec.pt/ghi/asrodrig/>

[5] Vijayan Sugumaran, A distributed Intelligent Agent-Based Spatial Decision Support System, Department of Business Administration, Le Moyne College, 1998. <http://www.isworld.org/ais.a.98/proceedings/geographic.htm>

PREVIEW

CLC Number: TP391

Foundation items:

Supported by the National Science Foundation of China (69999610) (60073016)

(1) (69999610) *“Digital Earth Demo System on High Speed Network”, sub item of “中国高速互连研究实验网NSFCnet” Project*

(2) (60073016) *“A study on an allied-agent approach for synergetic management of the heterogeneous spatial information systems in network environments”*

Biography:

Ding Li (1975-), male, M.S student, IEEE-CS student member. Research direction: software agent, GIS, distributed computing, software engineering,

Qu Lei (1975-), male, M.S student. Research direction: software agent, GIS, distributed computing, web applications.

Zhang Ying (1976-), male, M.S student. Research direction: software agent, GIS, distributed computing, metadata.

Luo Ying-wei (1971-), male, Assistant professor, IEEE-CS member, Ph.D. Research direction: software agent, distributed computing, GIS.

Wang Xiao-lin (1972-), male, Ph.D student. Research direction: software agent, GIS, Component technology.

Xu Zhuo-qun (1936-), male, Professor, IFIP/TC-3 member. Research direction: GIS, Decision Support System, Parallel compile environment.