Multi-Agent Environment MAGE

Bo Wang  Qijia Tian  Zhongzhi Shi

Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080

ABSTRACT

Agent-oriented programming is being thought as the next generation programming paradigm after the object-oriented programming. A multi-agent environment, MAGE, will be introduced in this paper, which is a platform for rapidly developing distributed, heterogeneous multi-agent systems. What MAGE provides includes a general running framework for agents, an agent description language (ADL), an agent communication language (ACL), an agent server and a set of CASE tools. The MAGE is written in Java and is a powerful tool to construct multi-agent systems for Internet applications.

Keywords: intelligent agent, multi-agent environment, distributed artificial intelligence

1. INTRODUCTION

An obvious trend in computer science developing is that the paradigm shift from algorithms to interaction [8][9]. Intelligent agent technology has been raised as the primary technology in the wave. Agent is the rational entity with big-granularity, high-intelligence and autonomy characteristics, and Multi-Agent system (MAS) is composed of a number of agents capable of communicating, coordinating, cooperating and even competing with each other. MAS attempt to simulate the rational behavior of human being by agent, to describe the system not through certain algorithm, but through rational interaction among agents. Intelligence agent is not only a kind of technology, but also a methodology. It provides a new approach to implement large-scale, distributed, self-adaptable and complicated software system. MAS are the focus of intense interest on the part of computer science and artificial intelligence. MAS are being used in an increasingly wide variety of applications [2][3][11] At the same time, the research of MAS has inherited many ideas and methods from other disciplines such as sociology, economics, management, and philology. And we have acquired some research efforts such as the contract net from economics[7] speech acts theory from philology[4].

Besides new concepts and frames for comprehending complicated system, agent technology also provide the approach to analyze, design and implement complicated software system. Agent and MAS have more and more attention in fields of both theoretical research and practical application. Researching agent-based software engineering, researching MAGE and developing agent-based application are being increasingly important. Shoham issued the programming paradigm of AOP [6]. In AOP, agent becomes the basic unit of software system, and the behavior of the whole system is decided by the interaction, cooperation and coordination of the agents making the system up. Agent-based programming is regarded as the alternative mainstream method of software design after the object-oriented programming [5][6][10][12].

MAGE, as introduced in this paper, is a tool for developing multi-agent system. It provides platform for rapid agent system development under network computing environment, and also the device to generate agents via ADL (agent description language) script. Users can generate agents with tailored specific attributes through versatile user interface (e.g. protocol selection). Thus, in the practical application environment, the system can automatically config agents according to the specific information from the users to meet the interaction needs with specific users. MAGE usually works in the highly distributed heterogeneous environment, and it is increasingly used on Internet platform and assembled with Web-based information system. Meanwhile, there are also many legacy systems that need an added agent shell to be re-used by the multi-agent system. MAGE is written in Java to guarantee the portability of the agents and interoperability under heterogeneous environment, and it also supports the agent-based distributed computing. As a function module, legacy systems and components facing a specific field can be connected to agents through function module-agent kernel interface. Then a agent is defined in ADL and co-works with other agents in multi-agent environment through AOL as communication language.

The structure of the remaining parts of this paper is that: MAGE structure introduction in the second chapter, MAGE-based multi-agent application system development in the third chapter, and conclusion in the last chapter.

2. The Architecture of MAGE

MAGE is mostly composed of four parts: Agent Running Frame, Agent Description Language, Agents Server, and Agent Communication Language.
2.1 Agent Running Frame

Agent running frame comprises ‘Agent Kernel’, ‘Kernel-Function Module Interface’, and ‘Communication Module’, which provide some basic abilities for agent, such as agent communication mechanism, meta-knowledge presentation, execute module, and interface to function module. As a general agent template, the agent running frame connected function module will be a real agent.

The structure of Agent in MAGE is shown in Fig1. Communicator communicates with environment directly, receiving and sending messages. ACL Parser is responsible for interpreting message received from Communicator, or packing data into message that agent send to outside world, and then deliver the message with specific format (such as ACL) to Communicator that sends it out. Agent kernel is used to manage Agent internal attributes, such as capabilities, acquaintance information, and behavior of Agent, etc. The mental states (mostly refer to intention) of Agent are represented by agent behavior. And agent capabilities are implemented by Function. By calling underlayer communication functions in function module, it realizes connection with outside world. Function Module Interface provides various methods to embed function, such as built-in, accessories, or dynamic link. Scheduler adjusts agent behavior based on agent mental states. ADL parser is used to read agent attributes definitions, including agent name, agent local address, agent acquaintances addresses (especially communication server address), agent capabilities, agent behavior (provided by session definition), and also extra-variable, and extra-object which relate to the working process of agent. Agent kernel captures agent attributes from ADL parser; produce agent behavior rules according to sessions in ADL, and then scheduler implements agent behaviors.

Function modules do implement agent capabilities. When agent wants to execute tasks or provide certain services for other agents, it sends a single call request to function module through kernel. In general agent module, the call request is sent by kernel-function module interface. Four primitives in the interface can be called by the kernel: GetResultFromFM(), SendResultToFM(), GetRequestFromFM(), SendRequestToFM(). The function module has corresponding primitives. Function module can be an object or an existing application, as called ‘legacy system’. For the class written by Java, kernel-function interface can be called directed. C/C++ objects also can be called through Java local methods. But it is complex to call legacy systems. In general, there are two methods to resolve the problem: One is to use socket to realize communication between agent kernel and function modules (for the prior systems or the systems hardly possible to change); the other is to encapsulate legacy systems into CORBA objects, then kernel-function interface can call the application through CORBA.

In MAGE, agents work according to following flow:

BEGIN
Initialize the internal parameters of agent;
Load ADL script and analyze it, then construct internal Database;
Load function modules, start corresponding threads;
Initialize communication module;
Send registration information facilitator (communication server);
REPEAT
Receiving Thread;
Check message queue;
If message queue is null, check again after a while;
Analyze message, and obtain the type of message;
IF (Message is system command) THEN
Message is performed directly by agent kernel;
ELSE
Put message into buffer until function module fetch it;
Sending Thread;
Encapsulate sending message with the format of ACL;
Send message through underlayer thread;
ENDREPEAT
END

2.2 Agent Description Language

Essentially agent is defined according to describe the role that agent acts as in MAGE, including agent address, capabilities, resource, working mode. Users write agent running scripts in ADL. Before running, agent-running frame reads script dynamically, allocate address and resource, and realize connection between agent kernel and function module. The Description of ADL includes agent name, local address, acquaintance addresses, capabilities, working mode, etc. The example of ADL script sees also figure 3. Due to the length of the paper, only capability and session are introduced here.

Agent capabilities are used to define capabilities that agent have and the services that agent can provide for outside world. Special function modules, thus corresponding to function module types, mostly offer agent capabilities agent capabilities that are divided into build-in, accessories, and dynamic link types. The elements of agent capabilities include the name, the type, the command line, and the source of agent capabilities. The name is the agent capabilities label. The type shows clearly which kind the agent capabilities belong to. The capabilities can be used through the command line. The source points to/out where the functions of the build-in and dynamic link capabilities come from, besides accessories.

The name, local address, acquaintance address, capabilities in agent script depict the characteristics of agent static states. The action of agent is follow the session. The agent has some sessions, which can execute in parallel. The session may be Java sentences, function module calling, or some special public functions. The
table below is the session part of auto-negotiation agent script.

```java
[Sessions]
BEGIN
Session
  if (performative.equals("query"))
    query();
End Session
Session
  if (performative.equals("negotiation_start"))
    priceNegotiation();
End session
END
```

2.3 Agent Facilitator

Facilitator is a kind of special and important agent. Facilitator maintains the database of all capabilities, states and addresses of agents in MAGE. There are some required services that facilitator, with the database, provided for other agents, including: (1) Name Service: agent that only has the agent name which it looks for can access the agent by making use of name service. (2) Query Service: Agent can get the services that other agents provide from facilitator. (3) Request Service: When the service agent need does not exist in current system, agent can apply to facilitator for request service. As soon as any agent can provide that service, facilitator will inform the agent who has applied the service. (4) Agent Life-Cycle Service: Facilitator has the function managing check-in and check-out of other agents, and it can start and kill any other agents.

2.4 The Structure of Communication

In the aspect of syntax, ACL in MAGE keeps compatible with KQML, a widely accepted message format and a message-handling protocol to support run-time knowledge sharing among agents, that ensure the openness of system and making the most of existing resources. At present, ACL has 16 primitives, which can be easily extended by users. It is noticeable that in a multi-agent system an agent can be a service provider as well as a client, so it is a symmetric distributed system.

The communication among agents is realized by the aid of facilitator in MAGE. For accomplishing self-goal or cooperating with other agents, agents must be capable of communicating each other. Thus, one of the most important tasks during agent creating is to tell facilitator the address and capabilities of itself in time. And then facilitator sets up the connection among agents.

The communication among agents is divided into two classes: the system class and the application class, both of which can be described by ACL. The communication between agent and facilitator is the system communication, including name service, query service, request service, and Life-Cycle service. The application communication often takes place among the common agents, which are to request service, confirm/deny service, query information and deliver results.

The next table shows the syntax of ACL:

3. MAGE-based Multi-Agent Application System

As a multi-agent environment, MAGE is the development platform of agent-based software engineering. We can construct an agent as the following process (shown in Fig2). After the requirement analysis, we get the script of agent and function module in design phase, and then we create the right agents with agent running frame reading agent script and connecting with corresponding function modules.

![Fig2. The Process of Building an Agent](image)

The task in requirement analysis is to analyze the roles and the relationship among the roles. During the phase of design, it is uppermost to build the modal of agent, and to establish a mapping from system to universal software architecture. Based on the result of design phase, we can ultimately set up the various ADL scripts of all agents of system.

An agent ADL equals the format criterion of MAGE, and it is very significant for correctly describing MAGE. Some tools are provided to help users build ADL scripts in MAGE. The development of multi-agent system looks like an application, and the project being correlative with the process of constructing multi-agent system is managed by an application manager, such as capabilities management, roles management, agents management, and script creating, etc. After we get capabilities, and config role through capabilities, and then config ADL script through role, the agent will be created finally. An example, to build ADL script by tool, is shown in Fig3.
We construct some multi-agent system using MAGE: Virtual Market, the agent-based electronic business platform, realizes auto-negotiation; WISE, the agent-based web intelligent information management system, realizes information collecting, analyzing and managing; the agent-based long-distance education system realizes active service during teaching process; and the agent-based swarm intelligence simulation system simulates the interactions between heterogeneous distributed units.

4. Conclusion

MAGE actively connects agent technologies and Internet/Intranet, realizes open soft bus and versatile soft components, and provides beneficial development environment for application system assembling. This system is of following features: (1) the system provides ADL to enable users to define the behavior model and build desired agents more conveniently. (2) the function module interface provides versatile connection ways for function joint, such as built-in, accessories, and dynamic link, etc. (3) It provides ACL to enable convenient interactive communication among agents. (4) It provides graphed interface to describe the knowledge and ability of agents and it employs work flow chart to show the multi-agent system criterion. (5) The soft component agent has autonomy, it can be both client and server, and it supports symmetric client/server work model. (6) The main application frame of MAGE provides effective ways to re-use software, and using agent development application system with high granularity and strong abilities, we can improve the development effectiveness of software and support the assembling of application system with better openness and flexibility. Many kinds of agents can then be assembled conveniently on Internet/Intranet and thus a unified environment for knowledge dealing, amount computation and data processing can be provided.

Reference


