Agent-based Grid Computing *[*]

Zhongzhi Shi, Mingkai Dong, Haijun Zhang, Qiujian Sheng

Key Laboratory of Intelligent Information Processing

Institute of Computing Technology, Chinese Academy of Sciences
PO Box 2704-28, Beijing, 100080,China
Email: shizz@ics.ict.ac.cn

ABSTRACT Establishing grids is an important undertaking in developing scalable distributed infrastructures. In this paper we have proposed a model for agent-based grid computing from the implement point of view. Based on the model agent-based grid computing system AGEGC has constructed by MAGE which is a multi-agent environment platform. AGEGC has applied to develop several application systems presented in the paper. We believe that AGEGC will be useful platform for research on semantic grid.

Keywords: Grid Computing, Multi-agent Environment, Agent-based Grid Computing System

1. INTRODUCTION

Research in "Grids" has become an area of active interest. The "Grid" is an emerging infrastructure that connects multiple regional and national grids to create a universal source of computing power—the work "Grid" was chosen by analog with the electric power grid, which provides pervasive access to power [Foster 1999] The origin of the term is believed to have been the CASA project which worked on linking supercomputing sites known as meta-computing [Catlett 92].The first generation grid systems involved solutions for sharing high performance computing resources. The Information Wide Area Year(I-WAY) project is a representative first generation grid system which the virtual environments, datasets, and computers used resided at seventeen different U.S. sites and were connected by ten networks. The I-WAY project was successfully demonstrated at SC 95 in San Diego and defined following applications: supercomputing, access to remote resources, virtual reality, and video, Web, GII-Windows.

Web Services provide a means of interoperability that was essential to achieve large-scale computation with a focus on middleware to support large scale information processing. In a Grid, the middleware is used to hide the heterogeneous nature and provide users and applications with a homogeneous and seamless environment by providing a set of standardized interfaces to a variety of services. We can regard this type of grid systems as the second generation, such as Globus [Foster 97], Legion [Grimshaw 97].

In order to build new grid applications the emphasis shifts to distributed global collaboration, metadata and service oriented approach which are three key characteristics of the third generation grid systems. There is a strong sense of automation in the third generation grid systems as follows:

- Containing detailed knowledge of its components and status;
- Constructing system dynamically;
- Seeking to optimize its behavior to achieve its goals;
- Being aware of its environment.

Agent-based computing is particularly well suited to a dynamically changing environment which is an important property for the third generation grid [Roure 2002]. The agent-based computing paradigm has following features:

- Autonomy - agent operate without intervention;
- Social ability – agents interact each other using an agent communication language;
- Goal driven – agent exhibit goal-directed behavior;
- Reactivity – agents perceive and respond to their environment.

Hence we can view the Grid as a number of interacting agents. In terms of the idea we have built a prototype of agent-based grid computing (AGEGC). The next section will discuss the four layer of model for Grid. Multi-agent Environment MAGE is introduced in Section 3. The key issues of AGEGC is discussed in Section 4. There are two applications, such as e-business, oil supply chain, which will be introduced in Section 5. Finally we will point out the future work.

2. A FOUR LAYER OF MODEL

From conceptual point of view a three-layer model for the Grid infrastructure was proposed by Jeffery in a strategy document in 1999 [Jeffery 99]. In this model the computing infrastructure consists of three conceptual layers: data/computation, information and knowledge. The data/computation layer is a lower layer which primarily concerned with computational and data resources. It is characterized as being able to deal with large-scale data, providing fast network and diverse resources as a single meta-computer. This layer builds on the physical grid fabric concerned with a distributed collection of files, databases, computers and devices.

The information layer means the information is represented, stored, accessed, shared and maintained. Here information is understood as data equipped with meaning. Uniform access to information sources relies on metadata to describe information

[*] This project is supported by High-Tech Programme 863 (2001AA113121) and National Natural Science Foundation of China (90104021).

and integrate heterogeneous sources. There are following functions in the information layer:

- Connecting together the major information sources
- Interfaces: homogeneous access to heterogeneous distributed information
- Sophisticated statistical analysis / reduction techniques for floating point numbers, textual information and multimedia information
- Special facilities for image processing, visualisation and virtual reality

The knowledge layer is concerned with the way that knowledge is acquired, used, retrieved and maintained. Here knowledge is understood as information applied to solve a problem and achieve a goal or decision-making. We can see that following functions will be contained in the knowledge layer:

- Acquire knowledge through KDD (knowledge discovery in database) technology of which a well-known component is data mining;
- Apply knowledge to support intelligent assists to decision makers;
- Provide interpretational semantics on the information.

Here, we have proposed a four-layer model for agent-based grid computing from the implement point of view:

(1) Common resources: consist of various resources distributed in Internet, such as mainframe, workstation, personal computer, computer cluster, storage equipment, databases or datasets, or others, which run on Unix, NT and other operating systems.

(2) Agent environment: it is the kernel of Grid computing which is responsible to resources location and allocation, authentication, unified information access, communication, task assignment, agent library and others.

(3) Developing toolkit: provide developing environment, containing agent creation, distributed computing, collaborative applications, problem solving, negotiation support, to let users effectively use grid resources.

(4) Application service: organize certain agents automatically for specific purpose application, such as power supply, oil supply e-business, distance education, e-government.

3. MULTIAGENT ENVIRONMENT

3.1 Introduction of MAGE

MAGE (Multi-AGent Environment) is an agent-oriented environment for software designing, integrating and running. It provides a new computing and problem-solving paradigm in terms of agent technology. MAGE has facilities to support agent mental state representation, reasoning, negotiation, planning, cooperation and communication. It provides system designing support, description and assembling of knowledge and capability, negotiation and cooperation designing for agent-based computing on the Internet. We spent more than ten years developing MAGE system, and now MAGE has reached Version 2.0.

MAGE is a software framework fully implemented in Java language. First it simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase; Second it simplifies integration of applications through multiple schemes of software reuses and an agent-oriented software design with a graphic user interface, and also it simplifies the running management through a powerful GUI with many run-time supportThe agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one when required. The system needs only Java Run Time version 1.2 or later version as a running environment.

MAGE is being used by a number of companies and academic groups, such as Legend , Academy of Electric Power and many others. Further details and documentation can be found at http://www.intsci.ac.cn/mage.

3.2 Work flow of MAGE

According to software engineering requirements we have developed a whole procedure for software developing and system integrating in terms of MAGE, which includes requirement analyzing, system designing, agent generating and system implementing. Fig 1 demonstrates the work flow of software system developed by MAGE.

3.3 MAGE platform architecture

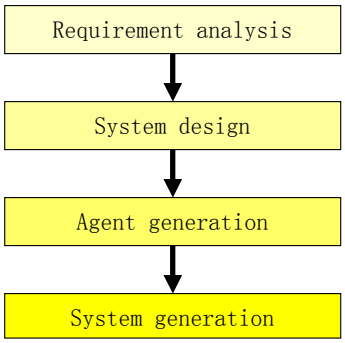

Fig 1. Workflow of MAGE

Fig 2 demonstrates the architecture of MAGE platform, which consists of the following components, each has specific capability set:

- **Agent Management System (AMS)** exerts supervisory control over access to and use of MAGE platform. It is the managing center of MAGE platform.
- **Directory Facilitator (DF)** provides yellow pages services to other agents, such as service registration, searching and updating
- **Message Transport Service (MTS)** is the default communication method between agents on different agent platforms.
- **Agent** is the fundamental actor in MAGE which combines one or more service capabilities into a unified and integrated execution model. Each agent can be designed with particular function for different applications.
- **Software** describes all non-agent, executable collections of instructions accessible through an agent. Non-agent software can be encapsulated with agent.

In addition, two auxiliary modules are provided to support designing agent systems: Agent Library and Function Components. User can compose different agents by using many kinds of function components provided by MAGE. Also user can select suitable agent class from agent library.

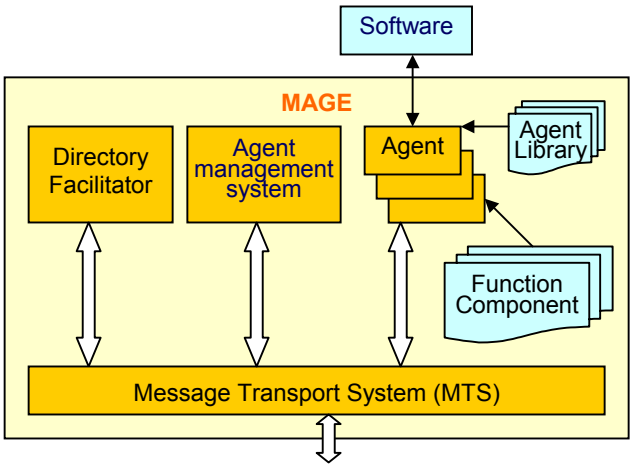

Fig 2. Architecture of MAGE platform

3.4 MAGE agent architecture

An agent in MAGE consists of six components: agent kernel, basic capabilities, sensor, communicator, function modules and knowledge base. Fig 3 demonstrates the detailed agent architecture.

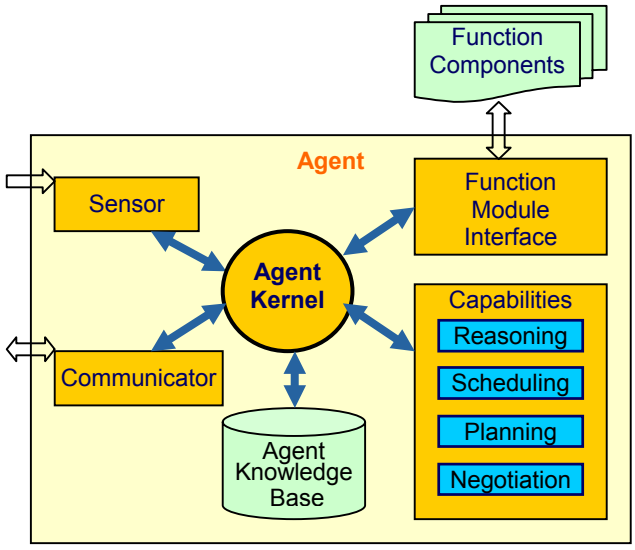

Fig 3. Architecture of MAGE agent

3.5 Agent life cycle in MAGE

Every agent in MAGE has a basic life cycle which includes six states: *Initiated*, *Active*, *Waiting*, *Suspended*, *Transit* and *Unknown*. And also MAGE support transition between different states, such as *Invoke* transform an agent from *Initiated* state to *Active* state. Refer to Fig 4 for detailed agent life cycle.

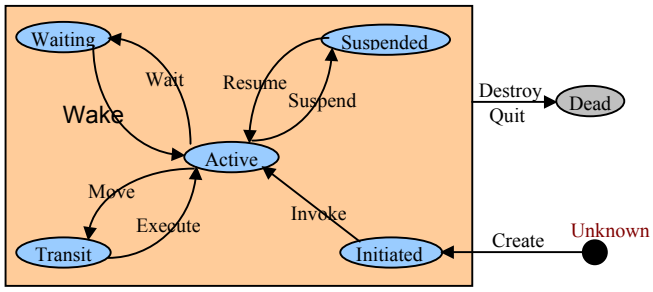

Fig 4. Agent life cycle in MAGE

3.6 Features of MAGE

3.6.1 Distributed computing platform

Distributed computing allows application program elements running on different hardware platform. MAGE is a distributed computing platform, and may be distributed on different host. MAGE is built on JAVA RMI and hides all implementation details from users, thus presents a unified computing environment for users. MAGE can support building distributed application easily.

3.6.2 Multiple schemes of software reuses

Software reuse can speed software development, reduce time to market and save resources, and so MAGE provides three kinds of method to reuse software (agent software or non-agent software coded in any language):

- **Embedded**
- **Executes from outside the system**
- **Dynamic link library (DLL)**

Through these methods, MAGE can provide software reuse in different granularity:

- **Reuse of application systems**
- **Reuse of sub-systems**
- **Reuse of components**
- **Reuse of functions***Multiple methods of agent generation*

MAGE provides three methods of agent generation:

- **Directly extends basic Agent class of MAGE**
  This method aims at building new applications from MAGE.
- **Agent Description Language (ADL)**
  ADL is used to describe the attributes (name, address, capabilities, etc) of an agent, and then MAGE can generate an agent according to this information and this agent has the corresponding attributes. This method aims at reusing software.
- **Clone**This is a way of agent generation at run-time.

3.6.4 Agent-oriented software design with a GUI

In specifying an agent system, we have found that it is highly desirable to adopt an external viewpoint. The description of an

agent system from the external viewpoint is captured in two models:

- **Agent Model** describes the hierarchical relationship among different abstract and concrete agent classes and identities the agent instances which may exist within the system their multiplicity and when they come into existence.
- **Interaction Model** describes the responsibilities of an agent class: the services it provides associated interactions and control relationships between agent classes. This includes the syntax and semantics of messages used for inter-agent communication and communication between agents and other system components: such as user interfaces.

According to the two models, MAGE provides a graphic tool for agent-oriented software design: XMIDE, it is shown in Fig 5.

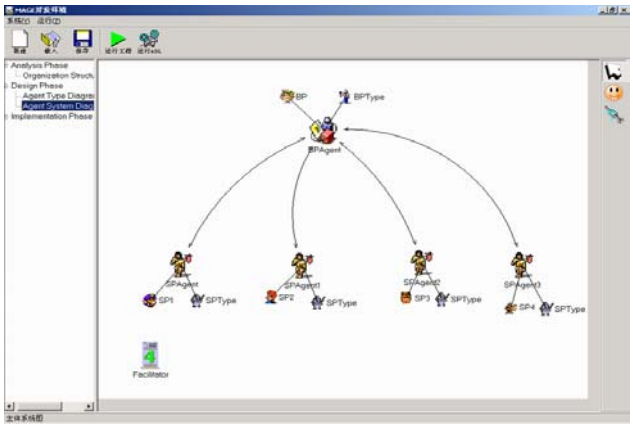

Fig. 5 GUI of XMIDE

industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-based applications. MAGE is not isolated, it is open and FIPA compliant, so it can communicate with any agent system complying with FIPA.

Agentcities is a worldwide initiative designed to help realize the commercial and research potential of agent based applications by constructing a worldwide, open network of platforms hosting diverse agent based services. The ultimate aim is to enable the dynamic, intelligent and autonomous composition of services to achieve user and business goals, thereby creating compound services to address changing needs. Up to now (2002-10-9) there are sixty agent platforms connected to agentcities.net. Our MAGE platform connected with agentcities successfully in 2002-4, and is the only agent platform connected with agentcities in China, and also become the second agent platform in Asia that connected with agentcities.

4. AGENT-BASED GRID COMPUTING SYSTEM AGEGC

4.1 The Architecture of AGEGC

From the four-layer grid model discussed in Section 2, we can see that how to provide an Agent Environment in the second layer is the key in implementing a prototype of this agent grid model because of the following reasons:

- First, Agent Environment *integrates* the components of Common Resources and makes these resources thus available and useful.
- Second, Agent Environment *provides* different kinds of agent grid common services for upper layers, thus upper layers can make use of Common Resources as easily as possible.

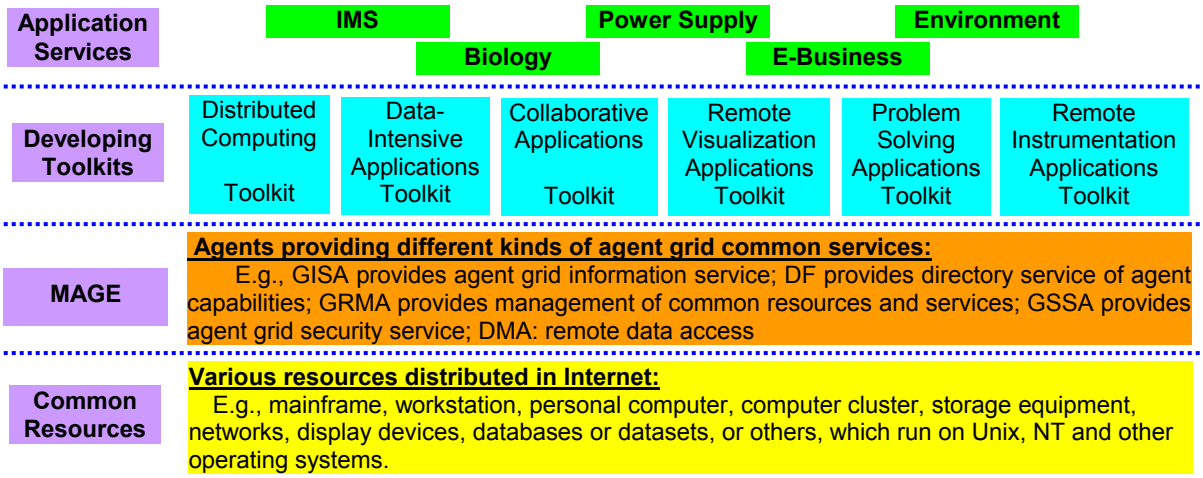

Fig 6. Architecture of AGEGC

3.6.5 Open and FIPA compliant

The Foundation for Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the

So we need a powerful agent environment that is competent for this. As we can see from Section 3, MAGE is a distributed agent environment and has many advantageous features. Based

on MAGE, we have implemented a prototype of agent gird: AGEGC. Fig 6 demonstrates the architecture of AGEGC.

In the following sections, several key issues of AGEGC will be discussed.

4.2 Directory Service

Grid applications often involve large amounts of data and/or computing and are not easily handled by today's Internet and web infrastructures. Grid technologies enable large-scale sharing of resources within groups of individuals and/or institutions. In these settings, the discovery, characterization, and monitoring of resources, services, and computations are challenging problems due to the considerable diversity, large numbers, dynamic behavior, and geographical distribution on the entities in which a user might be interested.

Consequently, directory services are a vital part of any grid software or infrastructure, providing fundamental mechanisms for discovery and monitoring, and hence for planning and adapting application behavior. In AGEGC, there are two types of directory service. Correspondingly, there are two types of agent: **DF** agent and **GISA** agent.

As we can see from the architecture of MAGE, **DF** (Directory Facilitator) is a mandatory component that provides a yellow pages directory service to agents. It is the trusted, benign custodian of the agent directory. MAGE may support any number of DFs and DFs may register with each other to form federations.

Every agent that wishes to publicize its services to other agents, should *register* its service description with DF. Also an agent can *deregister* itself form DF, which has the consequence that there is no longer a commitment on behalf of the DF to broker information relating to this agent. At any time, and for any reason, the agent may request the DF to *modify* its service description. An agent may *search* in order to request information from a DF.

We extend DF with a more abstract interface so that you can query: what agent A can do? Which agent is competent for a specific work? We have a reasoning machine embedded in DF, so it can "think"; if it thinks that a single agent can not do a specific job, it maybe return more than one agent whom together can do that job. And moreover, if it can't find agents that are capable, it may resort to other DFs.

GISA (Grid Information Service Agent) contains static and dynamic information about compute resources, as well as static and dynamic information about the network performance between compute resources. It provides information directory service. You query GISA to discover the properties of the machines, computers and networks or other common resources that you want to use: What is the state of the computational grid? What resources are available? How many processors are available at this moment? What bandwidth is provided? Is the storage on tape or disk? GISA provides middleware information in a common interface to put a unifying picture on top of disparate equipment.

The GISA uses the Lightweight Directory Access Protocol (LDAP) as a uniform interface to such information.

4.3 Resources Management

GRMA (Grid Resource Management Agent) is an important agent that provides capabilities to do remote-submission job start up. GRMA unites Common Resources and services, providing a common user interface so that you can finish a job with any common resource or service. GRMA is a general, ubiquitous service, with specific application toolkit commands built on top of it.

The GRMA processes the requests for resources for remote application execution, allocates the required resources, and manages the active jobs. It also returns updated information regarding the capabilities and availability of the computing resources to GISA and DF.

GRMA provides an API for submitting and canceling a job request, as well as checking the status of a submitted job. We extend Globus Resource Specification Language (RSL) to describe request. Users write request in ERSL and then request is processed by GRAM as part of the job request.

For example, if one wants to start a MAGE platform, but his own machine is busy. Then he first queries DF which agent has the capabilities to provide information services. Then DF tells him that GISA agent can do it. Then he queries GISA which machine is free in ICS-DOMAIN with a command *GISA-query-machine –lim ICS-DOMAIN*. After he gets an answer that *junez.ics.ict.ac.cn* is free, he can start a MAGE platform on host *junez.ics.ict.ac.cn* with a request *AGEGC-run junez.ics.ict.ac.cn "java mage.Boot –gui"*.

4.4 Data Management

In an increasing number of scientific disciplines, large data collections are emerging as important community resources. In domains as diverse as global climate change, high energy physics, and computational genomics, the volume of interesting data is already measured in terabytes and will soon total petabytes. The communities of researchers that need to access and analyze this data (often using sophisticated and computationally expensive techniques) are often large and are almost always geographically distributed, as are the computing and storage resources that these communities rely upon to store and analyze their data.

DMA (Data Management Agent) is an agent mainly aiming at remote data access. DMA provides basic access to remote data. Operations supported include remote read, remote write and append.

5. APPLICATION EXAMPLES

5.1 e-Business

We have developed a prototype of e-business. Figure 7 shows the architecture of agent-based e-business system. Four layers are constructed and each layer has its isolated functions.

The first layer includes all kinds of basic resources, such as database, models, and so on. Those resources are public and can be accessed by agent. For example, all data about the storage of e-business system are stored in a database, which is managed by a kind of DBMS such as Oracle or SQL Server. The database can be accessed by applications (agent) through standard SQL sentence.

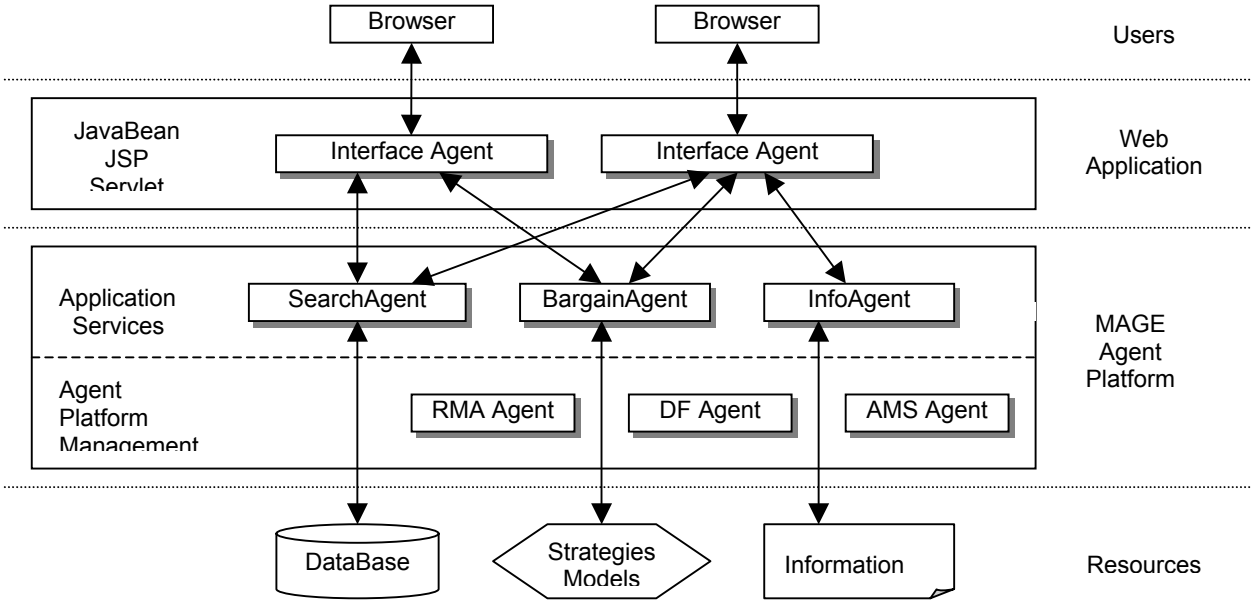

Fig 7. E-Business system

The second layer is MAGE agent platform, which is the core of the system. It can be divided into two sub-layers, agent platform management and application services. Agent platform management is bases of the MAGE platform. It is in charge of the management of agent platform and management of all agents' life cycle. AMS, DF and RMA are included in this sub-layer. Another sub layer is application services. It is constructed by many kinds of agents and each agent has it particular functions for given application, such as SearchAgent and BargainAgent. SearchAgent provides the service of database access. BargainAgent can employ different strategies to bargain with other buyer agent, so it can act as seller agent.

The third layer lies in Web server and it provides many kinds of Web application. It bases on the Web and establishes a friendly interface between users and application systems. In this layer, many services are provided through web pages and can be implemented by JavaBean, JSP, Servlet, and so on.

When a user login this web, an interface agent will be generated corresponding to the user. Also the interface agent will register in MAGE platform .If the user wants to search for some goods, the query information will be generated by the interface agent and be sent to the SearchAgent. After process the query, SearchAgent will send the search results to the interface agent. Finally, the user can get the results from web page which is generated by the interface agent.

Another interface agent will be generated if the user wants to buy something on the web. It acts as buyer agent. The user can setup the bargain strategies and models freely.

5.2 Oil Supply Chain

Military logistic system is one of the typical applications of multi-agent system technology. In this section, we present an Oil Supply Chain System (OSCS) which was developed with MAGE. In this prototype system, OSCS simply includes six agents, shown in Figure 8

LandUnitAgent is a consumer of oil, while other five member agents, with cooperation, are collectively responsible for providing oil to the consumer agent. In more details, LangOilAgent and NavyOilAgent are specific agents designed for Land Oil Base and Navy Oil Base of military respectively. LogisticHeadAgent is in charge of scheduling among oil consumer, oil providers, (i.e., Navy Oil Base and Lang Oil Base) and Oil Refinery which is acted by OilRefineryAgent. In the agent-based OSCS, TransporterAgent is delegated to transporter department which is responsible for transporting oil from the oil bases to the oil consumer.

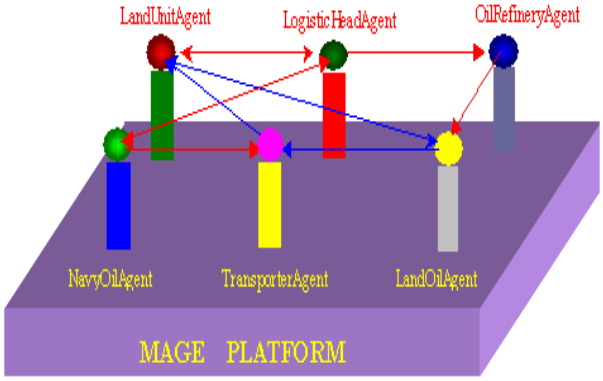

Fig. 8 Oil Supply Chain System

The OSCS agents are autonomous in the sense that they can perform different ACL acts to different outside agents in different contexts. For example, with the oil decreasing, LandUniAgent would send a REQUEST (ACL Message) to LandOilAgent for oil supply, when the oil storage goes under a pre-defined critical line. On receiving REQUEST message from LandUnitAgent, LandOilAgent would act variously, depending upon its current capability of oil. For example, if LandOilAgent can supply oil with the request volume, it would then send an

AGREE message to TransporterAgent in order to inform the real-world Transporter Dep. to transport the oil requested by LandUnitAgent. Otherwise, LandOilAgent would reply a REFUSE message to LandUnitAgent and initiate a REQUEST to LogisticHeadAgent for oil supply too.

Though autonomous agents can execute designed tasks with their own mental states and behavior rules, people can also access them by specific GUIs. Specifically, OSCS was provided with user interfaces by which people can view and control agents' lifecycle states, browse the transaction records, modify agents' behavior rules and so forth.

To sum up, the agent-based OSCS system can efficiently help human to manage the real-world OSCS system. In particular, intelligent agents can shorten the delay in practice, and ensure that the oil supply chain can work smoothly in the varying environment.

6. CONCLUSIONS

Establishing grids is an important undertaking in developing scalable infrastructures. In this paper we have proposed a model for agent-based grid computing from the implement point of view. Based on the model agent-based grid computing system AGEGC has constructed by MAGE which is a multiagent environment platform. AGEGC has applied to develop several application systems presented in the paper.

Due to the very generic nature of the grid computing, we can involve the research on it from different level, such as operating system layer, information layer, knowledge layer, service-oriented layer. Agent-based grid computing system AGEGC focuses on service-oriented layer in terms of current exist running environment. AGEGC will be useful platform for research on semantic grid.

7. REFERENCES

[1] A. Grimshaw, W. Wulf et al., The Legion Vision of a Worldwide Virtual Computer . Communications of the ACM, 40(1), 1997.

[2] C. Catlett and L. Smarr, Metacomputing, Communications of the ACM, June 1992, pp. 44-52.

[3] David De Roure, Mark A. Baker, Nick R. Jennings and Nigel R. Shadbolt, The Evolution of the Grid.

[4] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 11(2): 115-128, 1997.

[5] I. Foster and C. Kesselman (eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.

[6] Keith G. Jeffery, Knowledge, Information and Data, A briefing to the Office of Science and Technology, UK, February 2000.

[7] Zhongzhi Shi, Intelligent Agent and Applications. Science press, China, 2000