(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 January 2008 (17.01.2008)

PCT

(10) International Publication Number
WO 2008/006267 A1

(51) International Patent Classification:
G06F 17/40 (2006.01) *G06Q 30/00* (2006.01)
G06Q 20/00 (2006.01)

(21) International Application Number:
PCT/CN2007/000496

(22) International Filing Date:
13 February 2007 (13.02.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200610061542.5 5 July 2006 (05.07.2006) CN

(71) Applicant and
(72) Inventor: **LEE, Shu Tak, Raymond** [CN/CN]; Flat D, 13/F, Winfield Gardens, No. 34-40, Shan Kwong Road, Happy Valley, Hong Kong (CN).

(74) Agent: **SHENZHEN STANDARD PATENT & TRADEMARK AGENT LTD.**; Room 810-815, Yinzuo Int'l Building, No.1056 Shennan Boulevard, Shenzhen, Guangdong 518040 (CN).

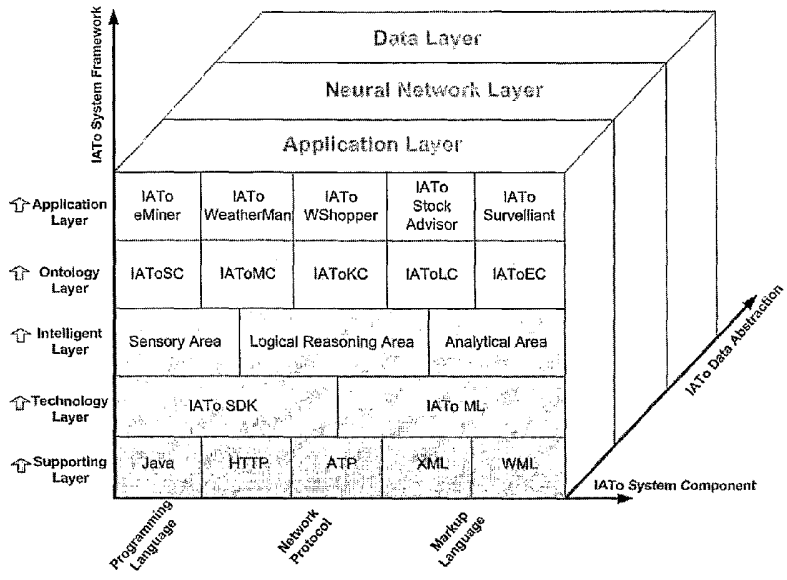
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** INTELLIGENT AGENT BASED DEVELOPMENT PLATFORM



(57) **Abstract:** An intelligent agent technology (IAT) based development platform comprises a first module which can provide full artificial intelligence and intelligent agent technology (IATology) based application interface for constructing various intelligent agent application and a second module for data storage and analyzing data intelligently as well as providing the analyzed results to the first module. An intelligent agent system with real sense, analysis and logical illation capability is provided. In addition an intelligent ontological agent based development environment (IATo) is provided to be the basic frame and development platform of the future intelligent E business.

WO 2008/006267 A1

INTELLIGENT AGENT BASED DEVELOPMENT PLATFORM

FIELD OF THE INVENTION

[0001] The present invention relates to artificial intelligent and information technology, particularly, relates to an intelligent ontological agent model as the basic framework and development environment for e-commerce applications.

BACKGROUND OF THE INVENTION

[0002] In information technology, especially intelligence and computer science field, intelligent agent is considered to a device which can recognize the environment by its sensors, and respond to the environment with its executive device. For example, referring to human body, eyes, ears and other sensory organs are the cognitive devices, while the executive devices are hands, legs, mouths and other parts of the body. When it comes to software, the cognitive and executive organs are encoded character streams.

[0003] The main purpose of artificial intelligence is to design intelligent agent programs, that is, to implement the mapping methods of cognition and action. This intelligent agent program has to operate on a certain computing device called framework. Said framework may be a common computer, or a special hardware tailored to perform certain tasks, or certain software between a computer and intelligent agent program for providing a certain degree of isolation, which enables programming in higher layers. In general, the structure enables the information as received by the sensors to transform as cognition, and provide feedback via execution program and hence produce a response.

[0004] Owing to the rapid development of e-Commerce and Internet technology in the recent years, many different e-Commerce applications and mobile computing systems have been operated in this cyberspace.

[0005] E-business is viewed as a big business opportunity as more and more people are focusing on the Internet. Numerous products are now available on the Internet, and product searching has become a burden for buyers. Meanwhile, sellers are difficult to locate target buyers and provide targeted promotion. It will be convenient to implement intelligent agent system, such that the agent system may actively search for online advertisements for buyers, go shopping online and even bargain for a better price, while the agent system simultaneously works for the seller to analyze different consumers' trends, and promote certain products to potential customers.

[0006] Therefore, in this 'sea' of information pool, the provision of an intelligent-based system (such as intelligent agents) seems to be a 'New Hope' in the future. However, contemporary agent-based developing environment such as IBM Aglet and ObjectSpace Voyager the provision of 'real' intelligent agent functionality is failed to support.

BRIEF SUMMARY OF THE INVENTION

[0007] We propose an innovative intelligent ontological agent-based development environment namely IATopia - "Intelligent Agent Utopia". The aim of IATopia is to provide comprehensive AI and ontological agent-based APIs and applications for future e-commerce and ontological-based applications.

[0008] The framework composes of two main model. The first module is "Application-Ontolgy-Intelligent-Technology-Supporting Layer" (AOITS) and the second module is "Data-Neural Network-Application Layer" (DNA)

[0009] The first module is a full artificial intelligence and ontology agent based application interface which constructs various intelligent agent application; and

[0010] The second module is used for data storage, intelligent data analysis process, and providing analyzed results to said first module.

[0011] The said first module comprises:

[0012] Application layer, comprises various intelligent ontology agent based application programs, said application programs are integrated by intelligent agent components of the intelligent layer and the data of said second module.

[0013] Ontology layer, base on the brain knowledge of agent, provides necessary knowledge for agent to initiate its logical and knowledgeable thinking. This layer of brain knowledge is named IATology-20000.

[0014] Intelligent layer, provides artificial intelligence basic base on the sense field, logical illation field and analysis field, while utilizes agent components provided by technology layer.

[0015] Technology layer, provides necessary mobile agent object application program interface for intelligent agent components of intelligent layer, comprises providing IATo SDK software development tools of full multi-intelligent agent based development platform, and providing understandable marking language to increase the communication of intelligent agent and IATo ML development tools of data transferring; and

[0018] Supporting layer, provides all necessary systems for supporting said layer, comprises programming languages, communication protocols and standardized file exchange format that are adopted to facilitate the development of the IATopia framework.

[0016] The said second module comprises:

[0017] Data layer, for storing raw data from intelligent agent brain. It is also the source of knowledge.

[0018] Neural network layer, for manipulating the data stored in the data layer so that knowledge can be generated and the thinking process can be initiated as

well as the agent can learn or correct itself according to its experiences; and

[0019] Application layer, with the fully support from the data layer and neural network layer, agent have enough knowledge and thinking capability to live and work autonomously.

[0020] The application programs in the said application layer comprise:

[0021] IATo eMiner, an intelligent ontological web-mining agent system for e-shopping.

[0022] IATo InfoSeeker, an intelligent ontological knowledge based information searching system.

[0023] IATo WeatherMAN, an intelligent ontological weather forecasting agent.

[0024] IATo WShopper, an integrated ontological intelligent fuzzy shopping agent for intelligent mobile shopping on the Internet.

[0025] IATo Stock Advisor, an intelligent ontological agent based stock prediction system.

[0026] IATo Surveillant, the automatic ontological agent based surveillance system.

[0027] The said intelligent agent comprises the following requirements: autonomous, mobile, reactive, proactive, adaptive, robust, communicative, learning, task-oriented, goal-driven.

[0028] The said IATo SDK development tools comprise all the mandatory components arranged on intelligent agent development platform. Said mandatory components comprise intelligent agent managing system, information transferring server and index arbitrator.

[0029] The basic function and run time properties of the said intelligent agent are defined by intelligent agent, lifecycle manager that provides all the control function, registration manager that records intelligent agent registration

information in the intelligent agent development platform, and communication manager that controls the message transfer in intelligent agent platform.

[0030] The said application program interface comprises: the first region that provides the functions of creating, activating, invalidating, copying, distributing, releasing and exiting, the second region that provides the functions of writing logs and displaying the information of activation in the server, the third region that provides tree type intelligent agent list, and the forth region that provides information broadcasting type.

[0031] The present invention provides an intelligent agent system with real sense, analysis and logical illation capability. In addition, an intelligent agent utopia (IATo) based development platform is provided to be the basic frame and development platform of the future intelligent E-business.

BRIEF DESCRIPTION OFTHE DRAWINGS

[0032] Figure 1 is the block diagram of the present invention.

[0033] Figure 2 is the block diagram of the user interface, in accordance with the present invention.

[0034] Figure 3 is the basic structure diagram of IATo SDK Intelligent Agent platform, in accordance with the present invention.

[0035] Figure 4 is the schematic diagram of Intelligent Agent communication, in accordance with the present invention.

[0036] Figure 5 is the schematic diagram of registration Intelligent Agent, in accordance with the present invention.

[0037] Figure 6 is the schematic diagram of creation Intelligent Agent, in accordance with the present invention.

[0038] Figure 7 is the schematic diagram of dispatch Intelligent Agent (sending end), in accordance with the present invention.

[0039] Figure 8 is the schematic diagram of dispatch Intelligent Agent (receiving end), in accordance with the present invention.

[0040] Figure 9 is the schematic diagram of the user interface, in accordance with the present invention.

[0041] Figure 10 is the schematic diagram of creation Intelligent Agent dialog box, in accordance with the present invention.

[0042] Figure 11 is the schematic diagram of dispatch Intelligent Agent dialog box, in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0043] The present invention disclosed a new intelligent ontology agent based development platform, called The Utopia of Intelligent Agent (IATopia) based development platform. The object is to develop a fully integrated intelligent ontology based multi-Intelligent Agent system, so as to be the basic frame and development platform of the future intelligent E-business.

[0044] The present invention may implement various Intelligent Agent based applications, comprising IATo TravelGuider and IATo Stock Advisor, etc.

[0045] In order to clarify the object, technical scheme and advantages of the present invention, various embodiments are described to provide detailed description of the present invention.

[0046] The framework of the present invention (functions and modules):

[0047] The present invention has fully integrated intelligent agent E-business application based intelligent ontology agent module. The system framework is shown in figure 1, unlike the Intelligent Agent system and Application Programming Interfaces (APIs) in the prior art, such as, IBM Aglets, ObjectSpace Voyager and IATo products. The present invention focuses on multi-Intelligent Agent communication and automatic operation. The object is

to provide full artificial intelligence and ontology agent based APIs, as well as future E-business application and ontology agent based application.

[0048] As shown in figure 1, the framework of the present invention comprises two main modules: The first module is Application-Ontology –Intelligent-Technology-Supporting Layer (AOITS) module, the second module is Data-Nerual-Network-Application Layer (DNA) module. AOITS module comprises application layer, ontology layer, intelligent layer, technology layer and supporting layer. DNA module comprises data layer, neural network layer and application layer. Various layers of AOITS are described in detail as follows.

[0049] Application layer: On the top layer of the system, comprises different intelligent ontology agent based application programs. Said application programs (IATo) are integrated by the intelligent ontology agent components from intelligent layer and data knowledge fields from DNA module. Various exemplary application programs are realized in this layer, which comprises:

[0050] IATo eMiner, an intelligent ontological web-mining agent system for e-shopping, comprises 1) IATo Authenicator-an automatic authentication system based on human face recognition, and 2) IATo Shopper- a fuzzy agent based Internet shopping agent.

[0051] IATo InfoSeeker, an intelligent ontological knowledge based information searching system.

[0052] IATo WeatherMAN, an intelligent ontological weather forecasting agent, which is the exten sion of previous research on mult-station weather forecasting using fuzzy neural networks. Unlike traditional Web-mining agents, which focus on the automatic extraction and provision of the latest weather information, IATo WeatherMAN possesses neural network based weather forecasting capability (AI services provided by the 'Conscious Layer' of the

IATopia module) to act as a 'virtual' weather reporter as well as an 'intelligent' weather forecaster for weather prediction.

[0053] IATo Shopper series, an integrated ontological intelligent fuzzy shopping agent with WAP technology for intelligent mobile shopping on the Internet.

[0054] IATo Stock Advisor, an intelligent ontological agent based stock prediction system using HRBFN (Hybrid Radial Basis Function Recurrent Network) for time series prediction.

[0055] IATo Surveillant, the automatic ontological agent based surveillance system.

[0056] Ontology Layer: based on the brain ontology knowledge of intelligent agent, provides necessary knowledge for intelligent agent to initiate its reasonable and knowledgeable thinking process. Said layer provides said ontology frame, that is the ontology centre, comprises the following 5 modules: Intelligent ontology based sensation centre (IAToSC), intelligent ontology based memory centre (IAToMC), intelligent ontology based knowledge centre (IAToKC), intelligent ontology based language centre (IAToLC), intelligent ontology based ethics centre (IAToEC).

[0057] Said five functional modules measure up the FIPA ontology service specification (XC00086D), an ontology agent (IATo Agent) will be designed and constructed with the following functions (namely, the "RATE" requirements):

[0058] Representation: to represent and discover public ontologies (especially ontologies in foreign platforms)

[0059] Administration: to maintain and administer the services and facilities provided by the iJAOS.

[0060] Translation: to communicate and translate concepts, meanings, and

universals between different ontologies and/or different content languages.

[0061] Explanation: To respond to and explain all queries concerning relationships between different concepts and ontologies.

[0062] Intelligent layer: This layer provides the intelligent basis of the IATopia system, using the agent components provided by the 'Technology layer'. The 'Conscious Layer' consists of the following three main intelligent functional areas:

[0063] Sensory area, for the recognition and interpretation of incoming stimulates, comprises: visual sensory agents using EGDLM (Elastic Graph Dynamic Link Model) for invariant visual object recognition, and, auditory sensory agents based on wavelet based feature extraction and interpretation technique.

[0064] Logic reasoning area, conscious and reasoning support, such as fuzzy angd GA (genetic Algorithms) rule based systems.

[0065] Analytical area, comprises various AI tools for analytical calculation, such as recurrent neural network based analysis for real-time prediction and data mining.

[0066] The technology layer: This layer provides all the necessary mobile agent implementation APIs for the development of intelligent agent components in the 'Conscious Layer'.

[0067] In the proposed latest version (v2.0) of the IATopia module, instead of IBM Aglets as the agent 'backbone', two innovative IATopia development tools have been developed, namely:

[0068] IATo SDK (IATopia Development Kit) and

[0069] IATo ML(IATopia Markup Language)

[0070] The main function of IATo SDK is to provide a comprehensive intelligent multi-agent based development platform with the provision for all

the intelligent agent-based Java classes and library, comprises: agents' communications, negotiations, intelligent agent tools, etc. The main function of IATo ML is the provision of a comprehensive markup language to enhance the intelligent agent communication and data exchange.

[0071] In this layer, server-side computing using Java Servlet technology is also adopted due to the fact that for certain intelligent agent-based applications, such as the IATo Shopper Series, in which limited resources (in terms of memory and computational speed) are provided by the WAP devices (e.g. WAP phones), all the IATo agents interactions are invoked at the 'backend' WAP server using Java Servlet technology.

[0072] The supporting layer: This layer provides all the necessary system supports to the 'Technology Layer', comprises:

1. Programming language support based on Java,
2. Network protocols support such as HTTP, HTTPS, ATP, etc., and
3. Markup languages support such as HTML, XML, WML, etc.

[0073] Each layer of DNA module is described in detail as follows.

[0074] Data layer: This layer is for storing initial data from agent brain and for IATology-20000 to generate knowledge, at this point, the thinking process may be initiated, and agent may learn or correct itself according to its experiences. This layer also explains why agent may think as a human.

[0075] Application layer: With fully support from data layer and neural network layer, the agent has enough knowledge and thinking capabilities to live and work on its own.

[0076] Further description of intelligent agent technology (IAT) is provided as follows:

[0077] Presently, most of the E-business systems on the Internet employ client/server manner. The disadvantages are: 1) High communication burden; 2)

reciprocity between enhanced and low level users. Certain objects are need to achieve said functions: Play the role of a human, to operate independently in the Internet, being autonomous, and to identify, process and solve problems on its own.

[0078] The definition of intelligent agent is: Intelligent agent (IA) is an example of intelligent in terms of device. IA may be a system, software program, program object or even a robot.

[0079] The intelligent agent in the present invention is with the following 10 basic requirements: 1) autonomous; 2) mobile; 3) reactive; 4) proactive; 5) adaptive; 6) robust; 7) communicative and cooperative; 8) learning; 9) task-oriented; 10) goal-driven. According to these basic requirements, the working theory of the IATopia agent in the present invention is described as follows:

[0080] IATopia Agent is a Java-based program. It is a sub-class of Thread class, so it can execute its life cycle asynchronously and concurrently inside the IATopia server. It implements the Serializable interface for packaging agent itself to migrate from host to host, and it also implements the Cloneable interface for copying itself to work concurrently with other instances of the agent. IATopia Server uses Java RMI as the transporting layer of IATopia Agents between IATopia hosts; agent will be serialized and sent to the target host by using RMI remote call.

[0081] IATo SDK framework in the implementation of IATopia:

[0082] IATo SDK is an intelligent agent development platform which implements FIPA Agent Management Specification utilizing Java 2 as the development language. The goal of IATo SDK is to provide an agent platform together with a set of API for simplify the development of agent system while ensuring the system is compliance to FIPA standard. The following table shows

the basic building blocks of the platform.

<table>
<tr><td colspan="2">Application Agents or Non-agent based user application layer</td></tr>
<tr><td>Agent Management Service</td><td>Directory Facilitators</td></tr>
<tr><td colspan="2">Agent Transport and communication System</td></tr>
<tr><td colspan="2">Java 2 Standard Edition (JDK 1.4)</td></tr>
</table>

[0083] To achieve this goal, IATo SDK offers the following features:

1. A FIPA-compliant Agent Platform with Agent Management System, Directory Facilitators and Message Transporting System. All these components are automatically started with the agent platform.
2. A registration manager to act as a directory facilitator to act as yellow page for registering or searching an agent inside the platform.
3. Message transporting mechanism for agents to communicate with each other and dispatching agents.
4. A lifecycle manager as an agent management system to control the agent's lifecycle within an agent platform.
5. A graphical user interface for the users to manage, monitor and log an agent's activities (Figure 2)

[0084] Basic component of IATo SDK: The IATo SDK agent platform is developed compliant with FIPA Agent Management Specification and includes all mandatory components that must be the in starting lineup with the agent platform, that are Agent Management System, Message Transport Service and Directory Facilitator. IATo SDK agent platform is developed by using pure Java 2 Standard Edition (JDK1.4). The mandatory components to startup IATo SDK

agent platform are LifeCycleManager, RegistrationManager and CommunicationManager. Figure 3 shows the basic architecture of the IATo SDK agent platform. IATo SDK provides the necessary mobile agent implementation APIs for the development of mobile intelligent agent systems. The basic functionalities and runtime properties of agents are defined by the Agent, LifeCycleManager RegistrationManager and CommunicationManager classes.

[0085] IIATo SDK provides necessary mobile intelligent agent object APIs for the development of mobile intelligent agent system. The basic function and run time property of intelligent agent are defined by Intelligent AgentPool, LifeCycleManager RegistrationManager and CommunicationManager.

[0086] Intelligent AgentPool: Basically, all agents must execute within a virtual place called AgentPool within the server. Thus, when an agent is created or dispatched, it must be put inside the AgentPool to start execution.

[0087] LifeCycleManager: LifeCyleManager act as the Agent Management System within the agent platform. It provides all functions of controls within the agent platform, comprises creating, suspending, resuming, dispatching and disposing agents.

[0088] Intelligent Agent: The Agent is the main character of the mobile agent concept. This is because it is a mobile software object that can transfer its software code and status from one host to another in order to perform a specific task. This can convince the development of a Code on Demand system. The agent has its own mechanism to broadcast or send messages to another agent for communication. When agent wants to ask for a specific service from another agent, firstly it will ask the RegistrationManager for the related agent's information and then ask the AgentPoolManager to get the reference of the wanted agent for further communication using the message passing mechanism

(Figure 4).

[0089] RegistrationManager: The RegistrationManager maintains a list of the registered agent's information including the agent's name, classes location and information about the service that the agent provides. It also protects the IATopia Server from anonymous attack. This is because an agent must be registered before it is allowed to execute inside the agent platform.

[0090] CommunicationManager: The message channel is maintained by the CommunicationManager. It controls the messages passing within the agent platform. It also provides a network communication channel for the agent to dispatch from local to remote sites.

[0091] Internal operation of IATo SDK (Data flow and input/output): By using these basic components, IATopia Server can provide a number of operations that helps mobile agent to perform their task.

[0092] Register Intelligent Agent: as shown in figure 5, Agents must be registered before they can *live* in the IATopia Server. Therefore, users are required to input the agent's information (e.g. Agent's name, code base and task description) by using the graphic user interface provided by IATopia Server (IATopia Server GUI). After receiving agent's information from user input, IATopia Server GUI will generate a request to the RegistrationManager. RegistrationManager then initialize an object RegistrationInfo and then save in the RegistrationTable. Finally IATopia Server GUI will be updated to inform the user after the registration is successful.

[0093] Create Intelligent Agent: as shown in figure 6, after registering an agent, user can create an agent by using the IATopia Server GUI, by choosing an appropriate agent's name in the list. The IATopia Server GUI will ask the RegistrationManager to get the basic code of the agent class file.

[0094] Then, IATopia Server GUI will send a request to the

LifeCycleManager to create an agent. After the agent file is loaded into Java VM, the LifeCycleManger will send the agent's reference to the AgentPoolManager, the AgentPoolManger will add the agent reference to the ActiveAgentPool. Finally IATopia Server GUI will be updated to inform the user after the creation is successful.

[0095] Dispatch Intelligent Agent: as shown in figure 7 and 8, when the agent is created within the IATopia Server, user can use the IATopia Server GUI to select and dispatch agent. When the IATopia Server GUI receives the user's request, it will forward the request to the LifeCycleManager. LifeCycleManager will ask the RegistrationManager for the AgentCodeBase and then AgentPoolManager for the AgentReference.

[0096] And then send a dispatch request to the CommunicationManager to dispatch the agent object and class file if necessary. There is a server listener in the remote machine. When the listener receives the dispatch request, it will forward the request to the LifeCycleManager. The LifeCycleManager will then check with the RegistrationManager that is that agent already registered in the remote server.

[0097] And then it will receive the agent object and then ask the AgentPoolManager to add the AgentReference into the ActiveAgentPool and update the IATopia Server GUI to notify the user that an agent has come to this server.

[0098] Programming IATo SDK: IATo SDK is developed with Java 1.4, since Java is an object oriented language. By using object oriented approach, program class or interface can be reused or further extend its function, so that the effect of outputting basic functionality may be saved. Java is a popular programming language; development of java program is cheaper and quicker then other programming languages. The portability of Java compiled code can

easily be migrated to different kinds of system. Finally, the built-in network supporting programming mobile intelligent agent is another advantage.

[0099] User interface: As shown in figure 9, there are 4 regions on the user interface. The upper part contains a set of button including Create, Activate, Deactivate, Clone, Dispatch, Dispose and Exit. These buttons provide a user-friendly interface for the user to control the lifecycle of an agent. The lower part of the interface is the system log that will show the activities that have been taken place inside the server.

[0100] The middle part of the interface is divided into two areas. The left area displays the lists of agents that are in different status tree style, while the right hand side displays in a message broad style.

[0101] A mobile agent must be registered before starting its activities. This feature is also a security protection of the mobile agent system because it can prevent anonymous agent entering the system to carry out expected damaging action. The registration of agent must be done by user by clicking the Create button. Then a "Create Agent Dialog" will pop up to ask for required information about the mobile agent (Figure 10).

[0102] User must input a valid class name of the agent and the source path that contains the class files inside the file system. Task description is an optional input, it is used to register the service in the RegistrationManager. Then the other agent can search for this agent by searching the registered service. After registering the agent with the task description and source path, user can activate the agent at any time to initiate its operation.

[0103] User can dispatch the mobile agent object to the other remote host at any time by clicking the Dispatch button. Then the "Dispatch Agent Dialog" will pop up to ask for information about where the mobile agent should be dispatched (Figure 11). Users are required to input the destination host name/IP

and the port number that the remote server is listening to.

[0104] Intelligent Agent Class: The Agent is the main character of the mobile agent concept. This is because it is a mobile software object that can transfer its software code and status from one host to another in order to perform a specific task. This can ensure the development of a Code on Demand system. The agent has its own mechanism to send messages to another agent for communication.

[0105] The Agent class is the basic class that the programmer can extend to create their own customized mobile agent. The API provides all functionalities that the agent can control its own lifecycle including the method for dispatching, deactivating, and disposing itself.

[0106] The dispatch method makes the agent hang the execution, save its status into a file and then send the status to the remote host. And to resume the execution code with the most updated status in the remote host. The deactivate method make the agent stop the execution. The dispose method will stop the agent thread's execution and also clear the status in the memory.

[0107] The Agent class also has a set of methods to get the attributes or the current status of the agent object. The getAgentName method can get the name of the agent inside the platform. The getAgentID method can get the ID assigned to the agent. The getStatus method can check the agent in active or inactive state.

[0108] Now let's see how do we program our agent with the Agent class. We should import the IATopiaserver.*, in order to include all of the libraries that supporting us to write our agent. Then we can define our own agent by extending the Agent class.

```
import IATopiaserver.Agent;
public class HelloAgent extends Agent {
    // implementation of the agent's method.
```

```
}
```

[0109] When an agent is created inside the agent platform, the platform must call the run method as default to start the execution of the agent thread. Therefore, we can write what is the default action that the agent must take by overriding the run method.

```
public void run() {
    // default action of the action when created.
}
```

[0110] The agent can dispatch itself to the remote host by simply using the host name / IP and the port number that the remote server is listening to.

```
public void dispatch(String host_name, int port_num);
```

[0111] When the agent arrived the remote server, the IATopia Server of the remote host will call the arrived method on default to resume the execution of the agent thread. Therefore, we should tell the agent what to do in the remote host by overriding the arrived method.

```
public void arrived() {
    // what action need to take when resuming the execution on the remote host.
}
```

[0112] When one agent wants to talk to the other agent, it must get the reference of the other agent from the LifecyleManager by using getOtherAgent method with the agent's name.

```
public Agent getOtherAgent(String agent_name);
```

[0113] After getting the reference of the other agent, the agent can communication with each other. By using the sendMessge method, we can send an object which is in any object type and containing any information to the other agent.

```
public void sendMessage(Object msg)
```

[0114] On the other hand, when the agent receive a message from the other agent. We can override the handMessage method to hand and give the appropriate response to the message.

```
public void handleMessage(Object msg) {
    // what action need to take for the incoming message.
}
```

[0115] Sometimes we may need to get the reference of the LifeCycleManager in order to do some action, for example, creating the other agent. We can use the getLifeCycleManager method to get the reference of the LifeCycleManager.

```
public LifeCycleManager getLifeCycleManager()
```

[0116] LifeCycleManager

[0117] LifeCyleManager act as the Agent Management System within the agent platform. It provides all functionalities of controls within the agent platform that include creating, suspending, resuming, dispatching and disposing agents.

[0118] After getting the reference of the LifeCycleManager, we can do some operations to control the lifecycle of the other agent inside the agent platform.

[0119] We can create a new instance or reactivate the agent object which is already be registered in the agent platform by using the agent class name.

public Agent activateAgent(String agentName, String activate)

[0120] By using the deactivateAgent method, the agent will stop its activity immediately and change the status to inactive.

public void deactivateAgent(String agentName)

[0121] The disposeAgent method will stop the activities of the agent immediately and remove the object from memory.

public void disposeAgent(String agentName)

[0122] When we need to talk to the agent by the client program, we can use the getOtherAgent of the LifeCycleManager to get the agent reference. So that we can send message to the agent and then inform the agent that what it may need to do.

public Agent getOtherAgent(String name)

[0123] Sometimes we may need to broadcast some message to all of the agent inside the agent platform. Therefore, we can call the getAllAgent to get an enumeration of agent reference to send message to every agent.

public Enumeration getAllAgent()

[0124] RuntimeAgent: Sometimes we can write an application that is not necessary to be initiated by using the IATopia Server interface. Therefore, we can use a static class method from the RuntimeAgent class to create a new instance of the agent object by using the server name, listening port number and then agent class name as the parameter.

public static Agent createAgent(String server, int port, String agentname)

[0125] Various embodiments are provided below:

[0126] (1) HelloWorldAgent. This example shows the simplest way to create an agent which only display the Hello World Message on a awt frame.

```
import IATopiaserver.Agent;
import java.awt.*;

public class HelloWorldAgent extends Agent {
    transient Frame my_dialog;   // transient means that this class will not be
    transfer during the disptach
     public void run() {
        message = "Hello World! I am " + getAgentName();
            my_dialog = new MyDialog(this);
            my_dialog.pack();
            my_dialog.resize(my_dialog.preferredSize());
            my_dialog.show();
     }
```

```
}

class MyDialog extends Frame {
    private HelloAgent agent = null;
    private Label msg = null;
    MyDialog(HelloAgent agent) {
        this.agent = agent;
        layoutComponents();
    }
    private void layoutComponents() {
        msg = new Label(agent.message);
        Panel p = new Panel();
        add(p);
        p.setLayout(new FlowLayout());
        add(msg);
    }

    public boolean handleEvent(Event ev) {
        if (ev.id == Event.WINDOW_DESTROY) {
            hide();
            return true;
        }
        return super.handleEvent(ev);
    }
}
```

[0127] (2) HelloWorldAgent2. This example is the same structure as

HelloWorldAgent but the agent will show the Hello World Message on a awt frame when it arrived the remote host.

```
public class HelloAgent2 extends Agent {
    transient Frame my_dialog;
    String message = null;

    public void run() {
        dispatch("IATopia1", 4444);
    }

    public void arrived() {
       message = "Hello World! I am " + getAgentName();
       my_dialog = new MyDialog(this);
          my_dialog.pack();
          my_dialog.resize(my_dialog.preferredSize());
          my_dialog.show();
    }
}
```

[0128] (3) In this example, an agent called TalkAgent will be created in 2 IATopia Server. The user need to click the connect button to make the connection between the two chatting agent in different host. Then the users can talk to each other by using the chatting interface. This examples shows that how can we create a new instance of the other agent (msgAgent) by using the getLifeCycleManager method. Also, this example also demonstrates how do the agent send and handle the message to do the communications.

```
import IATopiaserver.Agent;
public class TalkAgent extends Agent {
    transient Frame1 frame;
    String message = null;

    public void run() {
       frame = new Frame1(this);
       // codes to show the Frame1 interface
    }

    public void setDialog(Frame1 dlg) {
          this.frame = dlg;
    }

    public void handleMessage(Object msg) {
       if
(msg.toString().substring(0,msg.toString().indexOf('@')).equals("Connect")) {

        frame.appendText("Connected from " +
         msg.toString().substring(msg.toString().indexOf('@')      +      1,
msg.toString().length()));
       }
       else {
        frame.appendText(msg.toString().substring(msg.toString().indexOf('@')
        + 1, msg.toString().length()));
       }
```

```
    }
}

public class Frame1 extends JFrame {
    TalkAgent agent = null;
    msgAgent msgagent = null;
    String host = "";
    int port = 0;
    String chat = "";

    //Construct the frame

    void btn_Connect_actionPerformed(ActionEvent e) {
      try {
        InetAddress addr = InetAddress.getLocalHost();
        host = destHost.getText();
        port = Integer.parseInt(destPort.getText());
       msgagent =
      (msgAgent)agent.getLifeCycleManager().activateAgent("msgAgent",
              "ACTIVATE");
        msgagent.setMsg("Connect@" + addr.getHostName());
        msgagent.getLifeCycleManager().dispatchAgent(msgagent, host, port);
      }
      catch (UnknownHostException ex) {
      }
    }
```

```
    void jbtn_Send_actionPerformed(ActionEvent e) {
      msgagent = (msgAgent)
agent.getLifeCycleManager().activateAgent("msgAgent",
          "ACTIVATE");
      msgagent.setMsg("msg@" + msg.getText());
      msgagent.getLifeCycleManager().dispatchAgent(msgagent, host, port);
      msg.setText("");
      msg.updateUI();
    }

    public void appendText(String _msg) {
      chat += _msg + "\n";
      text.setText(chat);
    }
}

import IATopiaserver.Agent;
public class msgAgent extends Agent {
  String msg = "";
  public void arrived() {
    Agent agent = getOtherAgent("TalkAgent");
    agent.sendMessage(msg);
  }
  public void run() {
  }
  public void setMsg(String _msg) {
    msg = _msg;
```

```
    }
}
```

[0129] While implementing the present invention, base on the development platform of intelligent agent, the present invention is not limited in various embodiments described above. The present invention may expand to other application programs, as long as employing the intelligent agent based development platform to implement various application programs.

Claims:

1. An intelligent agent based development platform, said platform provides intelligent agent based development environment, said development platform comprises:

First module, full artificial intelligence and ontology agent based application interface are provided to construct various intelligent agent application; and

Second module, for data storage, intelligent data analysis process, and providing analyzed results to said first module.

2. The development platform in claim 1, said first module comprises:

Application layer, comprises various intelligent ontology agent based application programs, said application programs are integrated by intelligent agent components of the intelligent layer and the data of said second module.

Ontology layer, base on the brain knowledge of agent, provides necessary knowledge for agent to initiate its logical and knowledgeable thinking.

Intelligent layer, provides artificial intelligence basic base on the sense field, logical illation field and analysis field, while utilizes agent components provided by technology layer.

Technology layer, provides necessary mobile agent object application program interface for intelligent agent components of intelligent layer, comprises providing IATo SDK software development tools of full multi-intelligent agent based development platform, and providing understandable marking language to increase the communication of intelligent agent and IATo ML development tools of data transferring; and

Supporting layer, provides all necessary systems for supporting said layer, comprises programming languages, communication protocols and standardized file exchange format that are adopted to facilitate the development of the

IATopia framework.

3. The development platform in claim 1 or 2, said second module comprises:

Data layer, for storing raw data from intelligent agent brain. It is also the source of knowledge.

Neural network layer, for manipulating the data stored in the data layer so that knowledge can be generated and the thinking process can be initiated as well as the agent can learn or correct itself according to its experiences; and

Application layer, with the fully support from the data layer and neural network layer, agent have enough knowledge and thinking capability to live and work autonomously.

4. The development platform in claim 2, application programs in said application layer comprise:

IATo eMiner, an intelligent ontological web-mining agent system for e-shopping.

IATo InfoSeeker, an intelligent ontological knowledge based information searching system.

IATo WeatherMAN, an intelligent ontological weather forecasting agent.

IATo WShopper, an integrated ontological intelligent fuzzy shopping agent for intelligent mobile shopping on the Internet.

IATo Stock Advisor, an intelligent ontological agent based stock prediction system.

IATo Surveillant, the automatic ontological agent based surveillance system.

5. The development platform in claim 1, said intelligent agent comprises the following requirements: autonomous, mobile, reactive, proactive, adaptive, robust, communicative, learning, task-oriented, goal-driven.

6. The development platform in claim 2, said IATo SDK development tools

comprise all the mandatory components arranged on intelligent agent development platform. Said mandatory components comprise intelligent agent managing system, information transferring server and index arbitrator.

7. The development platform in claim 1, the basic function and run time properties of said intelligent agent are defined by intelligent agent, lifecycle manager that provides all the control function, registration manager that records intelligent agent registration information in the intelligent agent development platform, and communication manager that controls the message transfer in intelligent agent platform.

8. The development platform in claim 1, said application program interface comprises: the first region that provides the functions of creating, activating, invalidating, copying, distributing, releasing and exiting, the second region that provides the functions of writing logs and displaying the information of activation in the server, the third region that provides tree type intelligent agent list, and the forth region that provides information broadcasting type.

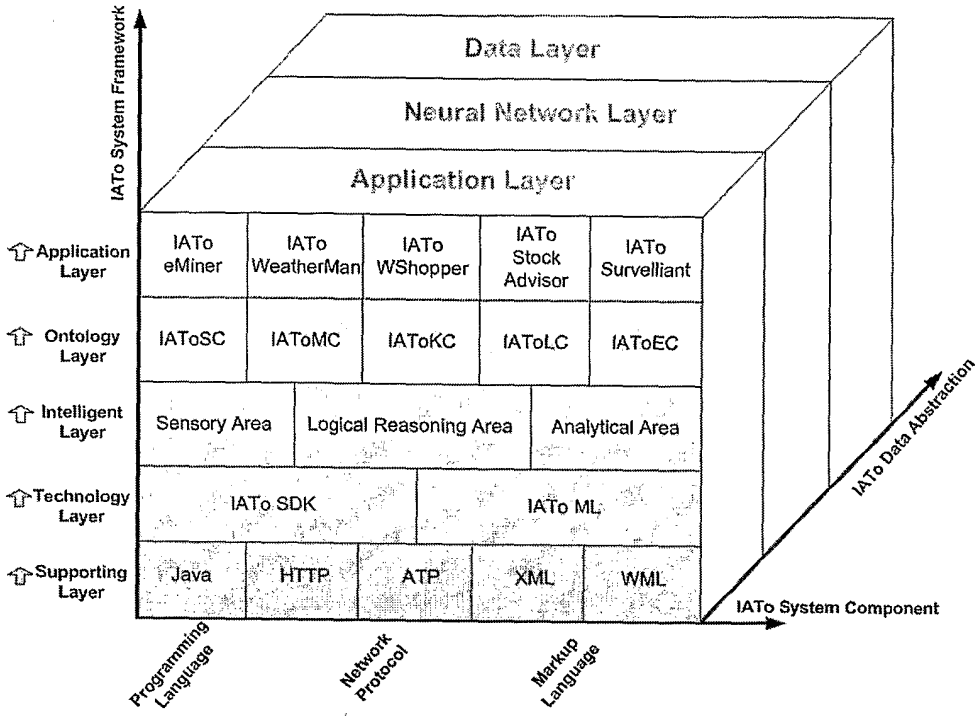


Figure 1

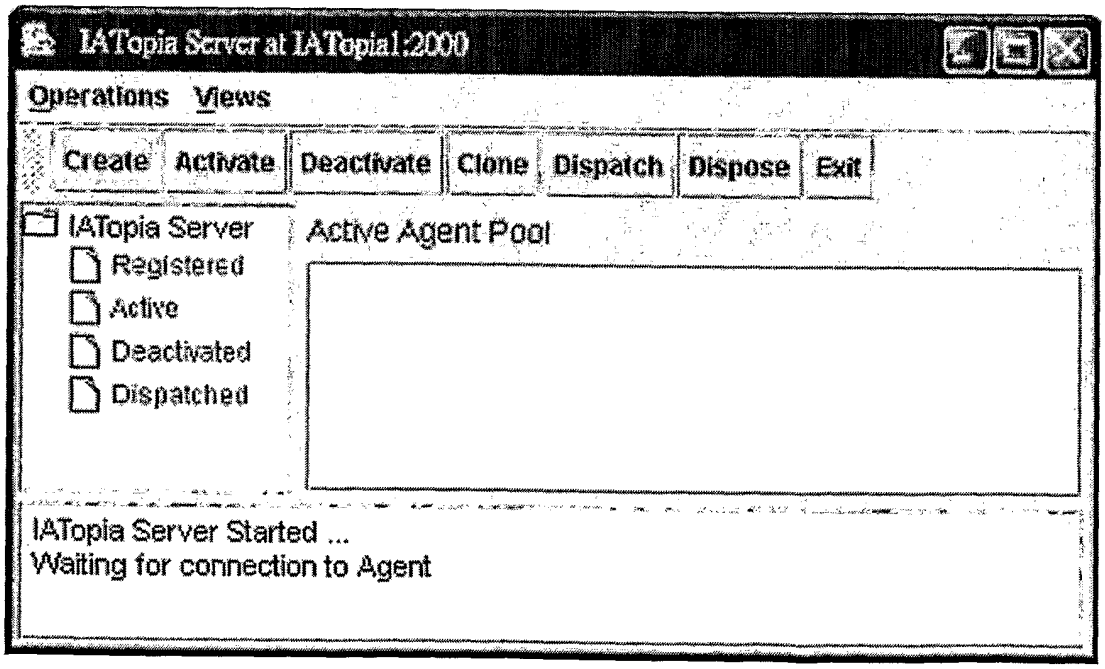


Figure 2

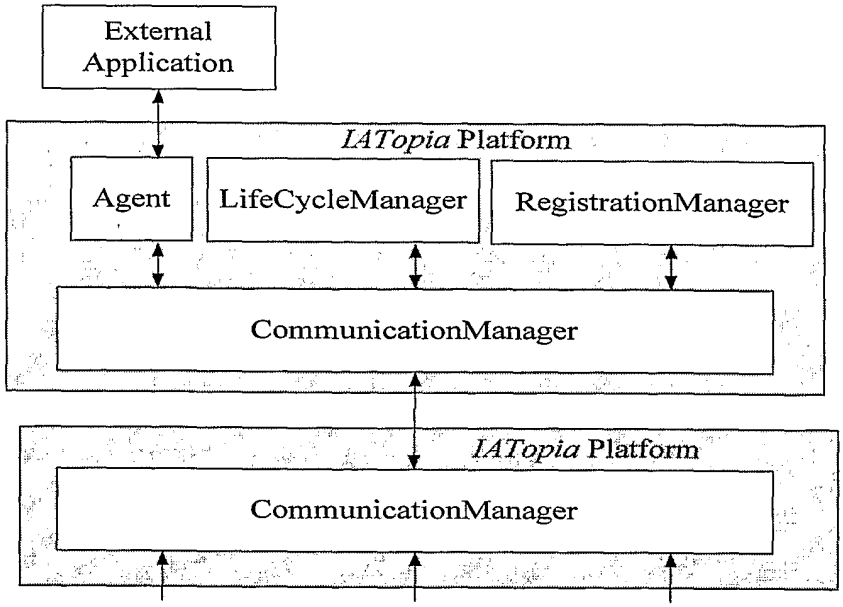


Figure 3

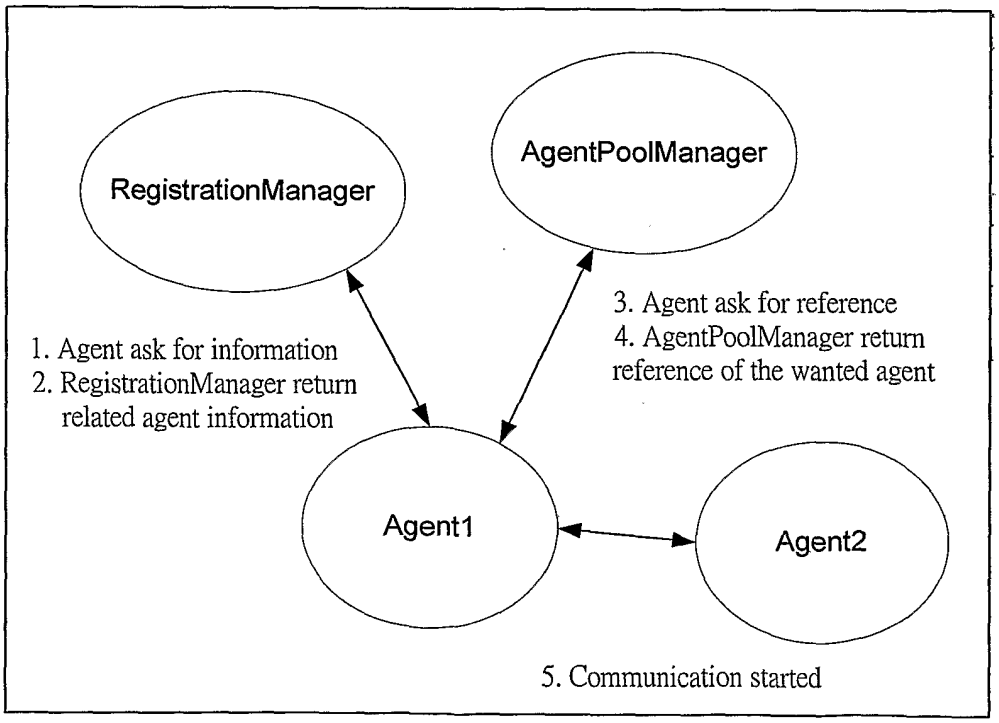


Figure 4

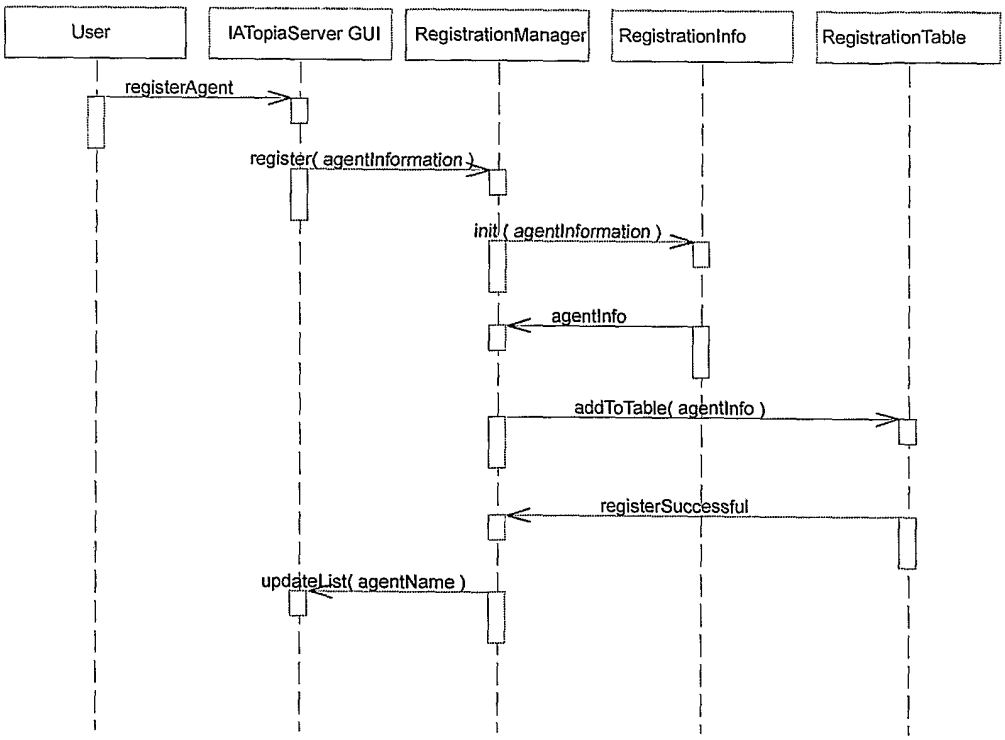

User
IATopiaServer GUI
RegistrationManager
RegistrationInfo
RegistrationTable
registerAgent
register(agentInformation)
init(agentInformation)
agentInfo
addToTable(agentInfo)
registerSuccessful
updateList(agentName)

Figure 5

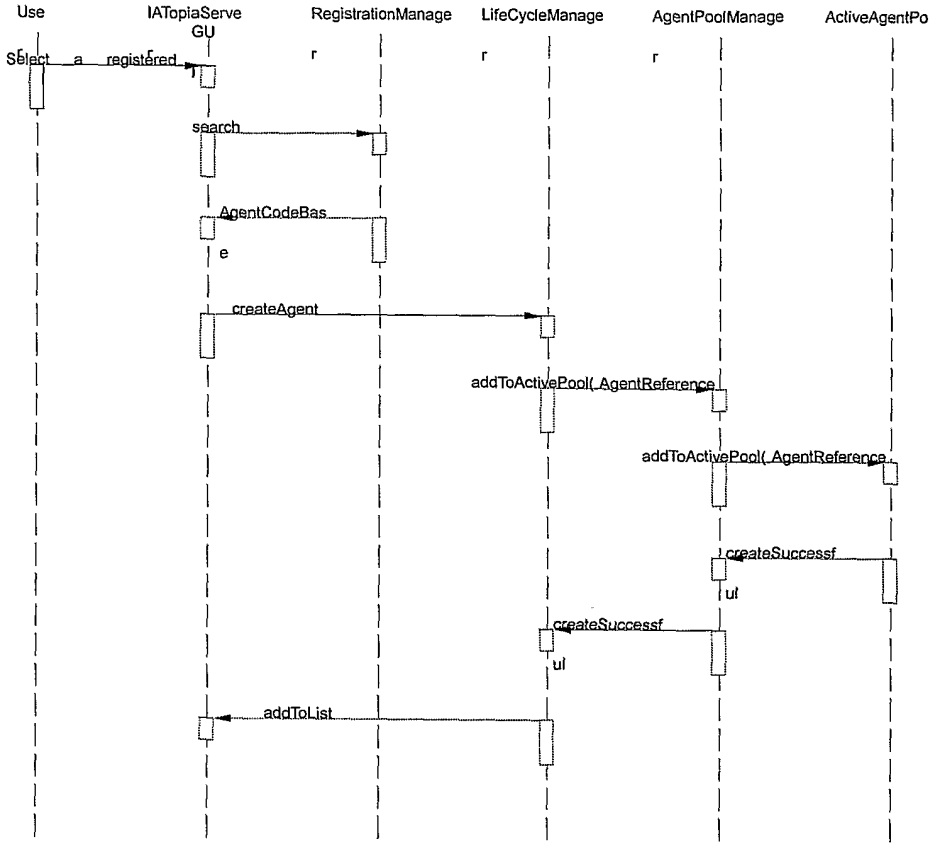

Use
IATopiaServe
GU
RegistrationManage
LifeCycleManage
AgentPoolManage
ActiveAgentPo
Select a registered
search
AgentCodeBas
e
createAgent
addToActivePool(AgentReference
addToActivePool(AgentReference
createSuccessf
ul
createSuccessf
ul
addToList

Figure 6

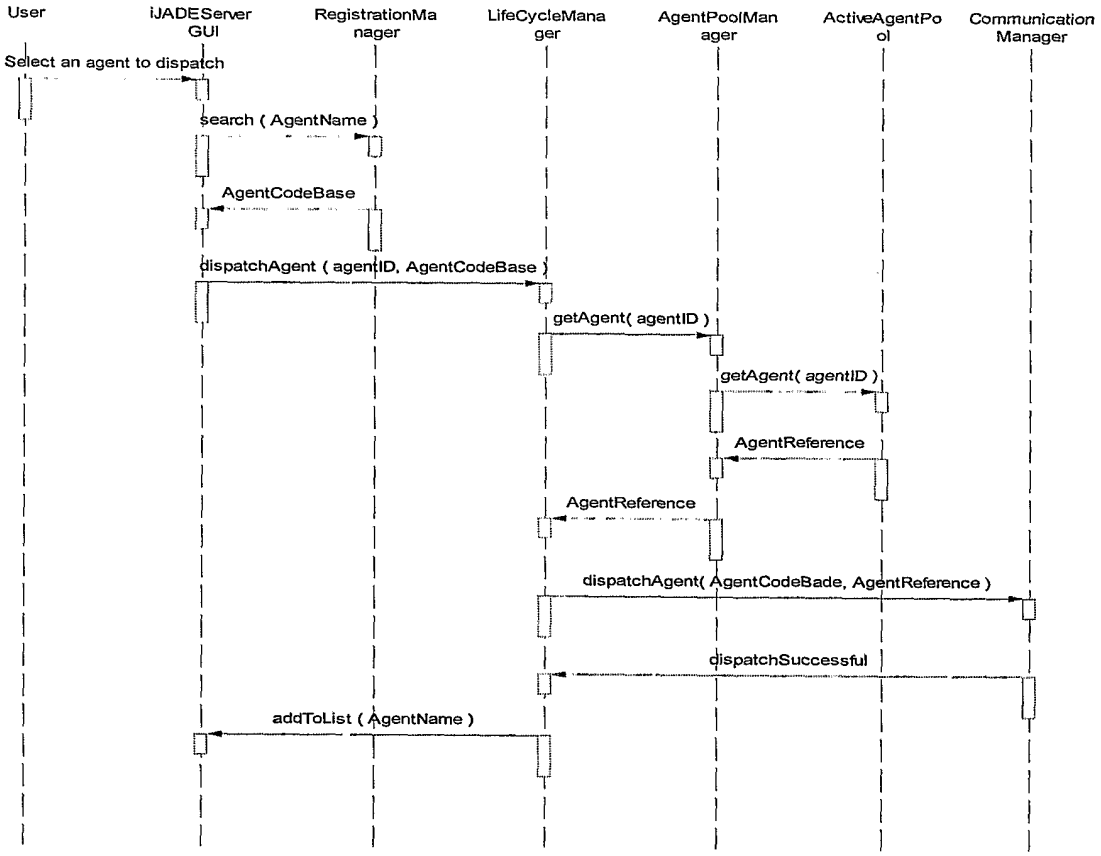

User
iJADEServer GUI
RegistrationManager
LifeCycleManager
AgentPoolManager
ActiveAgentPool
Communication Manager
Select an agent to dispatch
search (AgentName)
AgentCodeBase
dispatchAgent (agentID, AgentCodeBase)
getAgent(agentID)
getAgent(agentID)
AgentReference
AgentReference
dispatchAgent(AgentCodeBade, AgentReference)
dispatchSuccessful
addToList (AgentName)

Figure 7

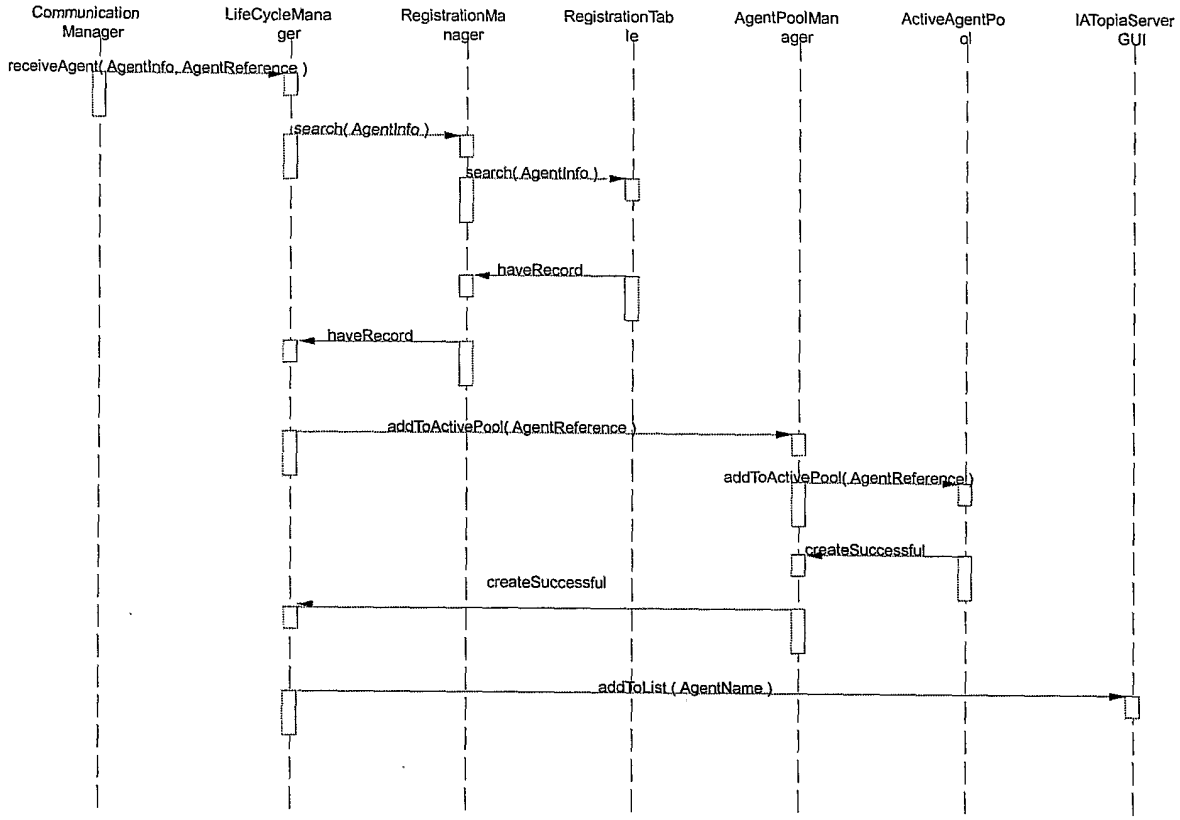


Figure 8

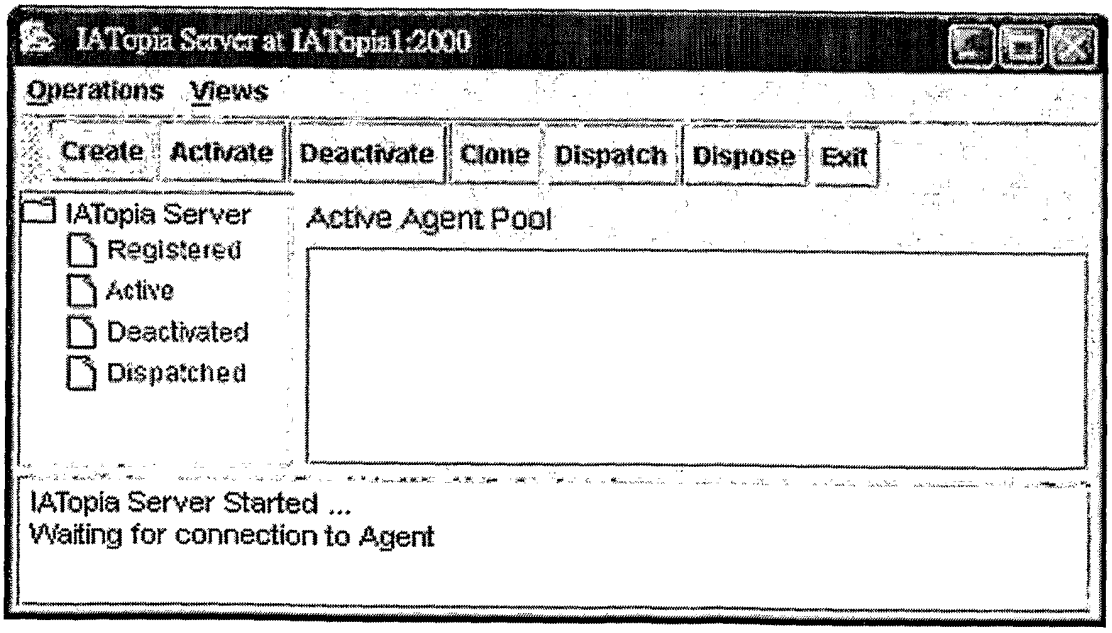


Figure 9

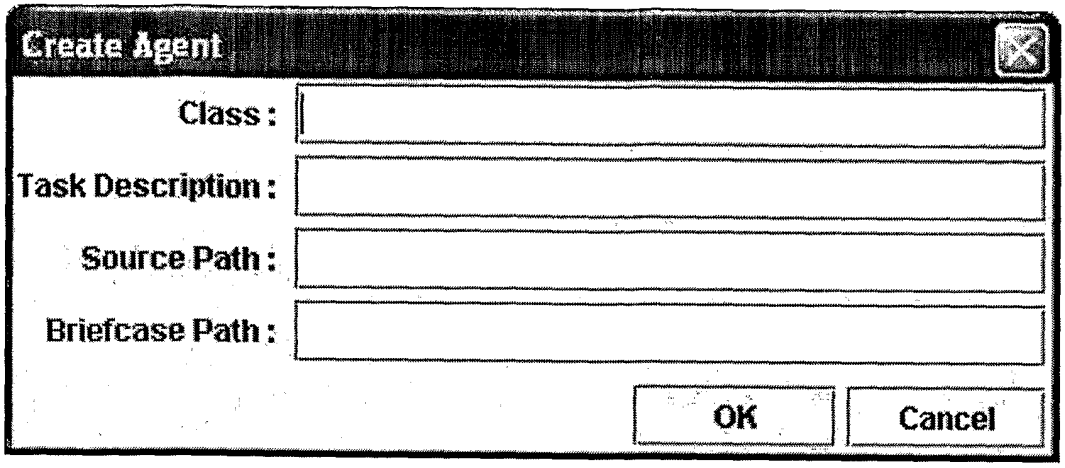


Figure 10

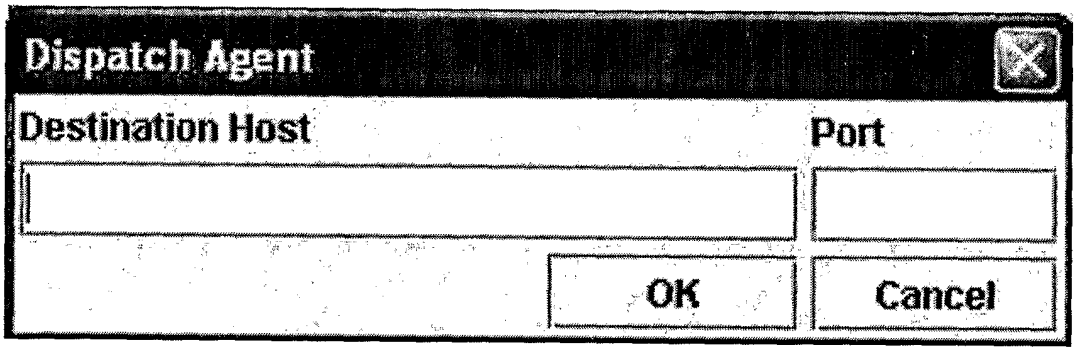


Figure 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2007/000496

A. CLASSIFICATION OF SUBJECT MATTER

See extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F G06Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI PAJ EPODOC CNPAT INTELLIGEN+ AGENT ARTIFICIAL STOR??? LAYER

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO01/26018A2 (U-KNOW SOFTWARE CORPORATION) 12 Apr. 2001(12.04.2001) page 1, line 10-page 11, line 30; figs.1-7	1-8
X	US2005/0144218A1 (Timothy James Heintz) 30 Jun.2005 (30.06.2005) paragraph [0068]-[0219]; figs.1-67	1-8
X	US2003/0084009A1 (Joseph Philip Bigus et al) 01 May 2003 (01.05.2003) paragraph [0011]-[0102]; figs.1-10	1-8
X	WO02/097588A2 (CAMELOT IS-2 INTERNATIONAL, INC. d.b.a. SKYVA INTERNATIONAL) 05 Dec.2002 (05.12.2002) page 1, line 22-page 25, line 30; figs.1-20	1-8

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&"document member of the same patent family

Date of the actual completion of the international search
27 Apr.2007 (27.04.2007)

Date of mailing of the international search report
21 · JUN 2007 (21 · 06 · 2007)

Name and mailing address of the ISA/CN
The State Intellectual Property Office, the P.R.China
6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China
100088
Facsimile No. 86-10-62019451

Authorized officer
CUI, Shangke

Telephone No. (86-10)82336309

Form PCT/ISA /210 (second sheet) (April 2007)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2007/000496

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
WO0126018A	12-04-2001	AU1493201A	10-05-2001
		US6381597B	30-04-2002
		TW501033B	01-09-2002
		CN1408093A	02-04-2003
US2005/0144218A	30-06-2005	none	
US2003/0084009A	01-05-2003	US2003084010A	01-05-2003
WO02097588A	05-12-2002	WO02069142A	06-09-2002
		US2002143653A	03-10-2002
		WO02099718A	12-12-2002

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2007/000496

CLASSIFICATION OF SUBJECT MATTER

G06F17/40 (2006.01) i
G06Q20/00(2006.01) i
G06Q30/00(2006.01) i