Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004)

A Platform to Integrate Well-Log Information Applications on Heterogeneous Environments

Chunsheng Li, Dan Cheng, and Chengqi Zhang, *Senior Member, IEEE*

Abstract— **The well-log information is very important for determining the crude oil and natural gas reservoirs. Because there are diversified generating methods and applications, well-log information is usually stored and processed on heterogeneous operating systems and data resources. How to share well-log information on heterogeneous environments efficiently is a big problem in petroleum industry. In this paper, we propose an agent-based platform in order to operate well-log information from heterogeneous data resources in uniform statements. The application and processing agents of well-log information on heterogeneous operating systems are dynamically integrated into the platform. A ring-based architectural model of middle agents, which is based on ring-based organizational structure and token-based coordination mechanism, is proposed for promoting the robustness and scalability of the platform. The heterogeneous data resources and operating systems are effectively integrated by using middle agents with ring-based architectural model. A prototype of this platform is implemented. The experimental results show that the platform works effectively.**

Index Terms—**Artificial intelligence, Heterogeneous platform, Software agent, Well-log information**

I. INTRODUCTION

Well-log information is a kind of geotectonic documents that are acquired by using specific instruments before the wells begin to produce oil. Well-log information is related to the crude oil and natural gas reservoirs because the data included in the information imply the rock properties, such as porosity and permeability. Nowadays, well-log information is being widely used in reservoir description, such as the classification of well log data into different lithofacies groups, facies-by-facies description of rock properties. With the deep applications of novel digital techniques in petroleum industry, the well-log information may be more used. But there are two impediments to use those data efficiently in the current computation environment. Firstly, because there are diversified generating methods with the development of measurement technology, well-log information is usually stored on different media, such as databases, data files, and parameter graphs. Parameter graphs have being digitized recently into data [8]. However, those data resources are heterogeneous. Secondly, applications based on well-log information are usually run on different computers and heterogeneous operation systems because of the various purposes. It is not thinkable that all applications share well-log information on heterogeneous resources. How to surmount above impediments is the goal of this paper.

Because agents are being advocated as a next generation model for engineering complex, distributed systems and will succeed as a mainstream software engineering paradigm [7], we propose an agent-based platform to integrate well-log information application on heterogeneous environment. The platform consists of three parts. Firstly, we employ practical well-log information processing algorithms, such as curve digitizing algorithms, to construct application agents. Those agents can work cooperatively to standardize well-log information on different media, and store them in databases or data files. Secondly, we wrap the databases and data file servers with middleware agents. Middleware agent is a wrapper of a specific database or data file server, which makes database or data file server exhibit the agent characteristics in agent-based environment. Lastly, a management mechanism of middleware agents and application agents is established based on middle agent idea. Middle agent provides intermediary service between requester agents and service provider agents. It is responsible for matching requester agents and service provider agents. In order to guarantee the robustness and scalability, we have designed a ring-based middle agent architectural model that is based on ring-based organizational structure and token coordination mechanism [2]. This ring-based middle agent architectural model effectively integrates heterogeneous data resources (different databases and different types of files) and heterogeneous operation systems (Unix and MS Windows). It becomes easy that new application agents using well-log information and middleware agents on heterogeneous environment can be dynamically integrated into the platform. The flexibility and robustness of middle agents with ring-based architectural model are better [2]. A prototype of this platform is implemented. The experimental results show that the platform works effectively.

The remaining sections of this paper are organized as follows: Section 2 chiefly presents the related work about this

C. Li is now with the Faculty of Information Technology, University of technology, Sydney, BROADWAY, NSW 2007 Australia (telephone: 61-2-9514 4534, e-mail: csli@it.uts.edu.au). He is also with the School of Computer Science and Engineering, Daqing Petroleum Institute, Daqing, Heilongjiang 163318 China.

D. Cheng is with the School of Computer Science and Engineering, Daqing Petroleum Institute, Daqing, Heilongjiang 163318 China (e-mail: chd_lily@163.com).

C. Zhang is with the Faculty of Information Technology, University of technology, Sydney, BROADWAY, NSW 2007 Australia (telephone: 61-2-9514 7941, e-mail: chengqi@it.uts.edu.au).

ICITA2004 ISBN 0-646-42313-4

research. Section 3 describes the framework of agent-based platform to integrate well-log information applications on heterogeneous environments. Section 4 discusses the architectural model of middle agents. Section 5 describes the prototype as case study. Section 6 concludes the paper.

II. Review of Related Work

Agent-based systems have shown much promise for flexible, fault-tolerant, distributed problem solving [7]. Much of the foundational work on agent technology has focused on methodologies to build agent-oriented systems, inter-agent communication protocols, and agent-based application systems [8]. Many agent-based models or frameworks were proposed for some specific applications. However, an integrated platform tailored to complex distributed systems, especially to integrated well-log information management, is rare.

About agent-based integration platform, the Open Agent Architecture (OAA) is a good example for integrating different agents in a system [4]. OAA made it possible for software services to be provided through the cooperative efforts of distributed collections of autonomous agents. Communication and cooperation between agents were brokered by one or more facilitators, which were responsible for matching requests, from users and agents, with descriptions of capabilities of other agents. The robustness of facilitators in OAA is a problem. Lipperts and Park developed an agent-based middleware to serve as a basis for user mobility [3]. An integrated environment acting as a software agent for discovering correlative attributes of data objects from multiple heterogeneous resources was presented in [1]. To reflect the standards of communication in a distributed heterogeneous system, a multi-level model and an agent template were proposed by Sheremetov and Smirnov [5].

How to design architecture is very important issue in agent-based system construction. Because agent-based system achieves its goal through interaction among agents, it is very important to design agent-based system so that agents can efficiently interact with each other. Managing the interaction is the goal of coordination: the space of agent interaction is no longer to be seen as merely the space of communication, but also the space of coordination [2]. Lee and Kim researched architecture modeling and evaluation for generic agent-based systems. They proposed seven valid architectural models for agent-based systems by combining three organization structures (Peer-to-Peer, Grouping with facilitator, and Tree) and four coordination mechanisms (Direct search, Matchmaker, Broker, and Contract-net). The idea of this research is valuable for designing agent-based system. However, the models are not enough to cover the gamut of work on agent-based system construction, especially, on the scalability and robustness of middle agent-based systems. All above work, to certain extent, promoted our research.

III. Framework of the Platform

The platform to integrate well-log information applications on heterogeneous environments is based on multi-agent technology. There are three kinds of agents in our platform, namely, application agent, middle agent, and middleware agent. Those agents will be organized by ring-based architectural model. All agents and their architectural model comprise a platform of well-log information management. The framework for the platform is shown in Fig. 1. The framework can be divided into three levels, namely, requester agent level, middle agent level and server provider agent level.

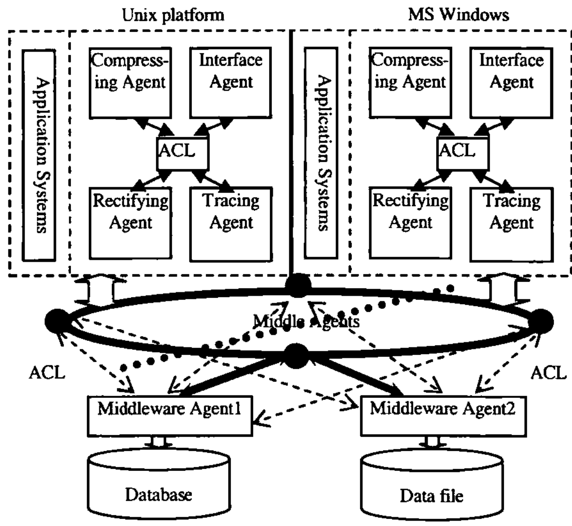


Fig. 1. Framework of the platform

Requester agent level mainly includes agents that are relevant to well-log information digitalization based on heterogeneous platforms and well-log information application agents. Well-log information digitalizing agents complete digitalizing well-log curves on parameter graphs. At the same time, they can standardize and process well-log information according to the user requests. Digitalizing process includes preprocessing (Interface Agent), compressing (Compressing Agent), tracing (Tracing Agent), and rectifying (Rectifying Agent). All agents contact with each other by Agent Communication Language (ACL), where KQML is employed in our implementation. In allusion to Unix and MS Windows platforms, two sets of agents are designed. If users need other applications, they can develop their own application agents and integrate them to the platform.

Middle agent level that is responsible for matching requester agents (application agents) and service provider agents (middleware agents) is made up of four agents (or more according to the number of platforms and applications). Those four agents are divided into two groups (or more groups), each of which includes two middle agents. One group is responsible for Unix platform and the other one is responsible for MS

Windows platform in Fig. 1. In each group, one middle agent acts as the host and the other one as duplicate. This design can guarantee the robustness of the platform. All middle agents keep the same information of service provider agents, where the information is synchronized by means of token ring.

Service provider agent level consists of heterogeneous data resources and middleware agents. When data resources and middleware agents change, middle agents must adjust their inner states according to environment. In order to operate databases and data files efficiently and effectively in agent-based environment, we design and implement agent-based middleware agents with legacy software wrapping technique to operate heterogeneous databases and data files in uniform agent communication language (ACL). From agent viewpoint, the middleware agent is a wrapper of a specific database or file server, which makes database or data file server exhibit the agent characteristics in agent-based environment. At the same time, a data manipulation agent communication language (KQML), which is suitable for both the databases or data files and agents, is employed.

A. *Request Agent*

There are two types of application systems, namely, well-log information application system and well-log information digitalizing system. Each system consists of several request agents. Well-log information application system includes request agents that are developed by users who want to use some well-log information. The platform is reserved an interface for development users to build their own applications.

Well-log information digitalizing system includes Interface Agent, Tracing Agent, Compressing Agent and Rectifying Agent. Those four agents complete scanning, preprocessing, compressing, and rectifying well-log information which is presented on graphs. Interface Agent is responsible for interaction between users and the system. User can control to scan well-log information graphs into computer as image files, select certain image files to preprocess with some suitable parameters so that different types of well-log information graphs are of the same characteristics in image files, and display results in various formats. Because each image file occupies about 80MB storage space, it is compressed with efficient algorithms before digitized. Compressing Agent deals with this work. Each image just occupies 1.7MB storage space after compressed. Tracing Agent use compressed image data to extract the data implied in the curves of well-log information graph. Rectifying Agent regards corrects mistakes of original graphs. The accurate well-log data are stored into database by means of middle agent and middleware agent.

B. *Middle Agents*

Middle agent is a kind of coordination organization of the intermediary service between requester agents and service provider agents. It is responsible for matching requester agents and service provider agents. According to present research results and different roles in a system, middle agent can be divided into Router, Broker and Matchmaker. Task of Router is similar to the function of route of the network. Router provides route support for requester agents to seek service provider agents so that requester agent can find essential service provider agent fast. Broker is an agent. It is responsible for managing the service provider agents and administers as an agent. When a requester agent needs service, it contacts Broker and gives task to this Broker. Then Broker submits the task to relevant service provider agent. After the service provider agent completes the assigned task, it will return results to Broker first. The Broker tells the result to the requester agent. Matchmaker is an agent too. It is responsible for providing the service provider agents with matching management. When requester agent needs service, it contacts with Matchmaker and submits the task to the Matchmaker. Matchmaker chooses one suitable service provider agent with relevant intelligent algorithms and notifies requester agent the service provider's address. Requester agent and service provider agent will get in touch directly and Matchmaker no longer intervenes their job.

According to the characteristic of every middle agent, we adopted Matchmaker technology to solve the problem of matchmaking of all agents in our platform. Middle agent maintains a registration table which records all information of the service provider agents, including service provider agent's address, port number, and service ability etc. The information comes from the registration information of service provider agents. When requester agent needs service, it sends a request to middle agent with parameters of requesting service type (related file, related database, etc.). Middle agent matches a suitable service provider agent by searching for the registration table according to the requesting parameters. If a service provider agent is matched, the middle agent will send the service provider agent's address and port number to the requester agent. Requester agent will contact the service provider agent directly with the address and port number, and finish the task. If an available service provider is not matched, the middle agent will send a message "no matching" to the requester agent.

C. *Service Provider Agent*

Service provider agent is also called middleware agent in our platform. Middleware agent is used for packing the non-agent legacy data resources, such as databases and data files. Non-agent legacy data resources will exhibit agent characteristics by wrapping with middleware agents. There are two aspects of task in each middleware agent. Firstly, it will advertise its registration information, such as its address, port number, and service ability, etc., to a certain middle agent for registering itself to the middle agent. Secondly, it will interact with requester agent by means of database-based or data file-based agent communication language, and complete the requests of requester agent. Each middleware agent wraps a specific database or the data file server with database or data file manipulation techniques, and completes various operations to database or data files. When the middleware agent completes its tasks, it will return a message "done" to the requester agent.

IV. RING-BASED ARCHITECTURAL MODEL

A. Architecture Modeling for Middle Agents

Architectural model for middle agents is based on organizational structure and coordination mechanism. Ring-based architectural model is powerful for promoting the scalability and robustness of the key components (e.g., middle agents) of agent-based system. The ring-based architectural model is based on logical ring organizational structure and token-based coordination mechanism. The logical ring organizational structure is shown in Fig. 2.

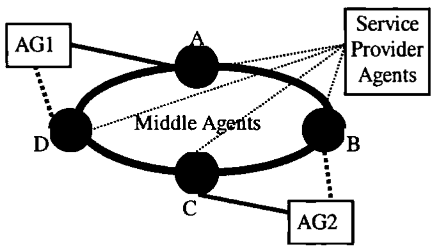


Fig. 2. Ring organizational structure

The middle agents A, B, C, and D serve for requester agents, that is, agent group 1 (AG1) and agent group 2 (AG2). Each agent group includes a set of requester agents that are of similar feature or close cooperate to achieve a chief goal, which can be regarded as an application. A requester agent in an agent group cannot directly contact with one in another agent group. However, a requester agent can belong to more than one agent group. All information of service provider agents is stored in every middle agent by token-based coordination mechanism. Each middle agent stores meta necessary information (like yellow page) about its supervised child agents to control the interactions among those child agents.

The middle agents are organized as logical ring structure. That is, A can only contact directly with B and D, but C. If A wants to contact with C, the message must be transmitted by B or D. For simplifying control mechanism, the logical ring is designed to be of the feature of one way in transmitting message. The middle agents are divided into two categories, namely, hosts and duplicates, according to their roles in the platform. The host middle agents, for example A and C, are the superior agents of agent group 1 and agent group 2, respectively, in normal situation. If the host middle agent A or C crashed, the ring would be automatically reorganized. At the same time, its duplicate middle agent D or B will act as a host and automatically proliferate a new duplicate middle agent. The meta information in host middle agent and its duplicate must be synchronized in time.

B. Coordination Mechanism

Coordination mechanism is another factor related to the interaction and organization among agents and middle agents. Without coordination, agents will not work in order because of the conflicts between them and cannot correctly response to service requests. The token ring coordination mechanism is designed for solving above problem. The token ring coordination mechanism consists of two algorithms, namely, logical ring establishing algorithm and token control algorithm.

For efficiently establishing the logical ring of middle agents, there is a coordinator among them at any time to establish a logical ring (system beginning, token lost or the ring broken down). In general, it does not matter which middle agent takes on this special responsibility, but one of them has to do it. For selecting a middle agent as coordinator, each middle agent is assigned a unique identify number (UIN; 0 to 255). In general, election algorithm attempts to locate the middle agent with the highest UIN and designate it as coordinator. Furthermore, it is assumed that every middle agent knows the UIN of every other middle agent. What the middle agents do not know is which ones are currently up and which ones are currently down. The goal of election algorithm is to ensure that when an election starts, it concludes with all middle agents agreeing on who the new coordinator is to be. In our research, we employed ring election algorithm [6]. This algorithm is based on the use of a ring, but without a token. It is assumed that the middle agents are logically ordered, so that each middle agent knows whom its successor is. When any middle agent notices that the coordinator is not functioning (the token interval is overtime), it builds an ELECTION message containing its own UIN and sends the message to its successor. If the successor is down, the sender skips over the successor and goes to the next member along the ring, or the one after that, until a running middle agent is located. At each step, the sender adds its own UIN to the list in the message. Eventually, the message gets back to the middle agent that started it all. That middle agent recognizes this event when it receives an incoming message containing its own UIN. At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (the list member with the highest UIN) and who the members of the new ring are. When this message has circulated once, it is removed and coordinator starts to manage the ring. The coordinator is in charge of managing the token, receiving service provider agent's advertisement, receiving new application-based system (agent group) request for proliferating a pair of new middle agents (host and its duplicate) for this new system, receiving a application-based system removing request for canceling the related middle agents (host and its duplicate) and removing them from the ring, maintaining he logical ring (control to proliferate new middle agents or remove old ones), etc.

When the ring is initialized, the coordinator will produce a token. The token circulates around the ring. It is passed from middle agent k to k+1 (modulo the ring size, 256) in point-to-point messages. When a middle agent acquires the token from its neighbor, it has three aspects of tasks to do. Firstly, middle agent checks the token if there is a service provider agent that has just advertised it to the coordinator. If yes, the middle agent registers service provider agent information. Secondly, if there is any maintaining message (new application-based system information, canceled an application-based system information, applied UIN, etc.) on the token, the middle agent updates related items or states. Thirdly, if the middle agent is a host, it checks if its duplicate is normal (If not, it proliferates a duplicate and makes it on the ring). If the middle agent is a duplicate, it checks if its host is normal (If not, it acts as the host, proliferates a duplicate and makes it on the ring). When the coordinator holds the token, it will clear the token and reload it again according to the environment.

C. *Architecture of the Middle Agent*

Middle agent completes four aspects of tasks. Firstly, it matches requester agent's request with service provider agent. Secondly, it may act as the coordinator. Thirdly, it looks after its duplicate or vice versa. The last, it receives and transmits the token on the logical ring. For achieving above tasks, a matchmaker consists of eight components: Communication Protocol Module (G PORT and C PORT), Coordinator Processing, Ring Manager, Token Manager, Service Provider Table, Status indicators, Matching, Control Module. The architecture is shown in Fig. 3.

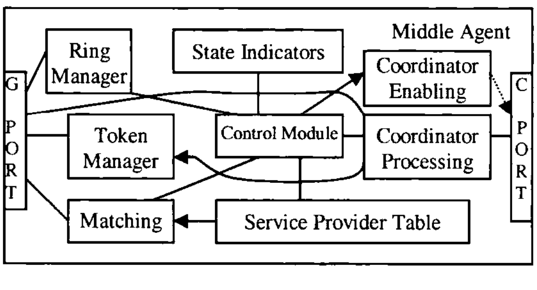


Fig. 3. Architecture of middle agent

Communication Protocol Module deals with the interaction with other agents by means of agent communication language (KQML). In the lowest tier, there are two communication ports, one for General purpose (G PORT) and another for Coordinator purpose (C PORT). All middle agents use the same port number to C PORT for convenient registration of new application-based systems and service provider agents, but just one middle agent is available at any time. If the middle agent acts as the coordinator, the C PORT is available; otherwise it is prohibited. Coordinator Enabling maintains this function. Coordinator Processing deals with the coordinator election, generating a token when the ring is initialized, managing current systems, proliferating a pair of new middle agents for the new application-based system, removing a pair of middle agents for the canceled system, and managing the registration of service provider agent. When coordinator holds the token, it puts the registration information of service provider agent in the token. Other middle agents will get the information from token and save it to the Service Provider Table. If a service provider agent is cancelled, the related item will be removed from the Service Provider Table. Ring Manager deals with changing states of the middle agent (from duplicate to host), proliferating a duplicate and adding it on the ring. There are two indicators to indicate its successor and predecessor on the ring. Token Manager deals with the receiving token, doing all tasks when a middle agent holds the token as described in last subsection, updating Service Provider Table and State indicators in accordance with the information on the token, transmitting the token to its successor. Service Provider Table holds the information about service provider agents. State indicators include middle agent state indicator, current system indicator, and the list of allocated UIN. Matching Module matches requester agent's request with a suitable service provider agent. Control Module, which is the control center of the middle agent, makes other components work in order.

V. Case Study

The middle agents with ring-based architectural model are applied in the application-based petroleum information services [8]. A prototype of this platform is implemented as shown in Fig. 4. In this platform, we developed two middleware agents (M1 and M2) with legacy software wrapping view to operate Oracle database (DB) and data file server (DFS). Those middleware agents are based on Oracle 7.3.4 database and FTP server. The SunOS Release 5.5.1, CDE 1.0.2, C programming development environment and PRO*C pre-compiling tool are required. The Socket programming is used for the low-level communication tool in developing the middleware agents.

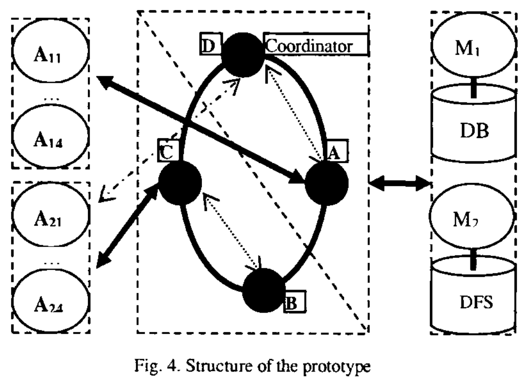


Fig. 4. Structure of the prototype

We develop two independent well-log curve-digitizing systems (one for Unix platform and another for MS Windows platform). Each well-log curve-digitizing system consists of Interface Agent (A_{i1}), Compressing Agent (A_{i2}), Tracing Agent (A_{i3}), and Rectifying Agent (A_{i4}). Host middle agent A directly serves for System1. Middle agent D is the duplicate of A and acts as the coordinator at the moment. Host middle agent C directly serves for System2. Middle agent B is the duplicate of C.

Middle agents automatically construct themselves to satisfy the environment of the middleware agents and current application-based systems. At the same time, they coordinate the interactions between application agent and middleware agent. The interactions are divided into three categories: middleware agent registration, new application-based system registration, and matching application agent with middleware agent. Fig. 5 presents the interaction pattern.

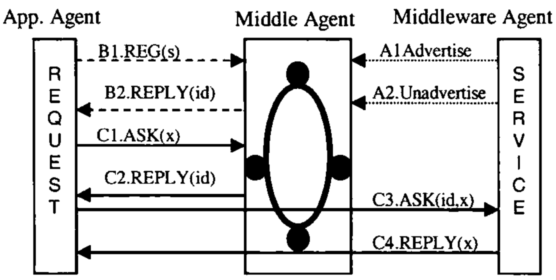


Fig. 5. Interaction between agents

When a middleware agent wants to register for providing service to application agents (App. Agents), it advertises its capability and feature to the coordinator (A1 in Fig. 5). When the coordinator holds the token, it registers the middleware agent information, puts the information on the token, and informs all other middle agents to register the middleware agent information as it did. When a middleware agent does not want to provide service to any agent, it sends 'unadvertise' to the coordinator (A2 in Fig. 5). The coordinator will remove its information and inform all other middle agents to remove the information as it did. If a middle agent detects a crashed middleware agent, it will ask coordinator to do the similar work as 'unadvertise'.

When a new application-based system wants to register for constructing its middle agents (host and duplicate), it sends its information to the coordinator (B1 in Fig. 5). When the coordinator holds the token, it registers the new application's information, proliferates two middle agents for the application and makes them on the ring, puts the information on the token, and informs all other middle agents to register the application information as it did. When the token comes back, coordinator clears this information from token and replies the new application with related middle agent's access port number (B2 in Fig. 5). The application-based system informs all its agents of the middle agent's port number. When an application does not want to run forever, it sends 'remove' to the coordinator. The coordinator will remove the related middle agents from the ring and informs all other middle agents to remove the system information as it did.

For matching application agent with middleware agent, the steps are as follows:

1) Application agent asks its related middle agent to answer to its request (C1 in Fig. 5).
2) Middle agent searches the capability and feature of each middleware agent, selects a middleware agent which is able to answer to the request, and replies the middleware agent's information to application agent (C2 in Fig. 5).
3) Since application agent knows which middleware agent is able to solve its request, it directly asks the middleware agent to answer the request (C3 in Fig. 5).
4) The middleware agent completes the request and replies the result to the application agent (C4 in Fig. 5).

After matching application agent with middleware agent, middle agent does not intervene any interaction between task agent and middleware agent, that is, Step 3 and 4 can be repeated for many times for other similar tasks.

VI. Conclusion

The platform to integrate well-log information applications on heterogeneous operation systems and data resources is available and efficient to manage difference application agents. It provides a uniform mode for reusing well-log information. However, the implementation of the platform still has some problems. For example, the matchmaking function of middle agents is very weak; there are some problems in the registration of new application-based system. Although the implemented prototype proves that the approach is available, some implementation must be improved.

References

[1] M. Chen, Q. Zhu and Z. Chen, "An integrated interactive environment for knowledge discovery from heterogeneous data resources," *Information and Software Technology*, 43(8), pp. 487-496, 2001.

[2] C. Li, C. Zhang, Q. Chen, and Z. Zhang, "A Scalable and Robust Framework for Agent-Based Heterogeneous Database Operation," in Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC2003), Vienna, Austria, pp.260-271, Feb. 2003.

[3] S. Lipperts and A.S. Park, "An agent-based middleware – a solution for terminal and user mobility," *Computer Networks*, 31(19), pp. 2053-2062, 1999.

[4] D.L. Martin, A.J. Cheyer, and D.B. Moran, "The open agent architecture: a framework for building distributed software systems," *Applied Artificial Intelligence*, 13(1-2), pp. 91-128, 1999.

[5] L.B. Sheremetov and A.V. Smirnov, "Component integration framework for manufacturing systems re-engineering: agent and object approach," *Robotics and Autonomous Systems*, 27 (1-2), pp. 77-89, 1999.

[6] S. Tanenbaum, *Distributed operating systems*, Prentice-Hall, Inc. New Jersey, USA, 1995.

[7] M. Wooldridge, *An introduction to multiagent systems*, John Wilery & Sons, Ltd. England, 2002.

[8] C. Zhang, C. Li and Z. Zhang, "An agent-based framework for petroleum information services from distributed heterogeneous data resources," in Proceedings of the 9th Asia Pacific Software Engineering Conference (APSEC 2002), Gold Coast, Australia, pp. 593-602, Dec. 2002.

Second International Conference on Information Technology & Applications

Conference Organising Committee

Conference Chair Prof Shi Guangfan Heilongjiang University, China

Conference Vice-Chair Dr. Fu Yuzuo Shanghai Jiao-tong University, China

Technical Chair Dr. Sean He University of Tech Sydney, Australia

International Advisor Dr Dapeng Tien Charles Sturt University, Australia

ICITA 2004 is organised by:

Heilongjiang University, Harbin, China

http://www.hlju.edu.cn

And supported by:

Shanghai Jiao Tong University, Shanghai, China

http://www.sjtu.edu.cn

IEEE, NSW Section, Australia

http://ewh.ieee.org/r10/nsw/

IEEE, CS Chapter, Beijing, China

http://www.cie-china.org/ieee-beijing/

Saora Inc., Japan

http://www.saora.com/

ICITA 2004 Harbin, China http://www.icita.org

Second International Conference on Information Technology & Applications

ICITA 2004

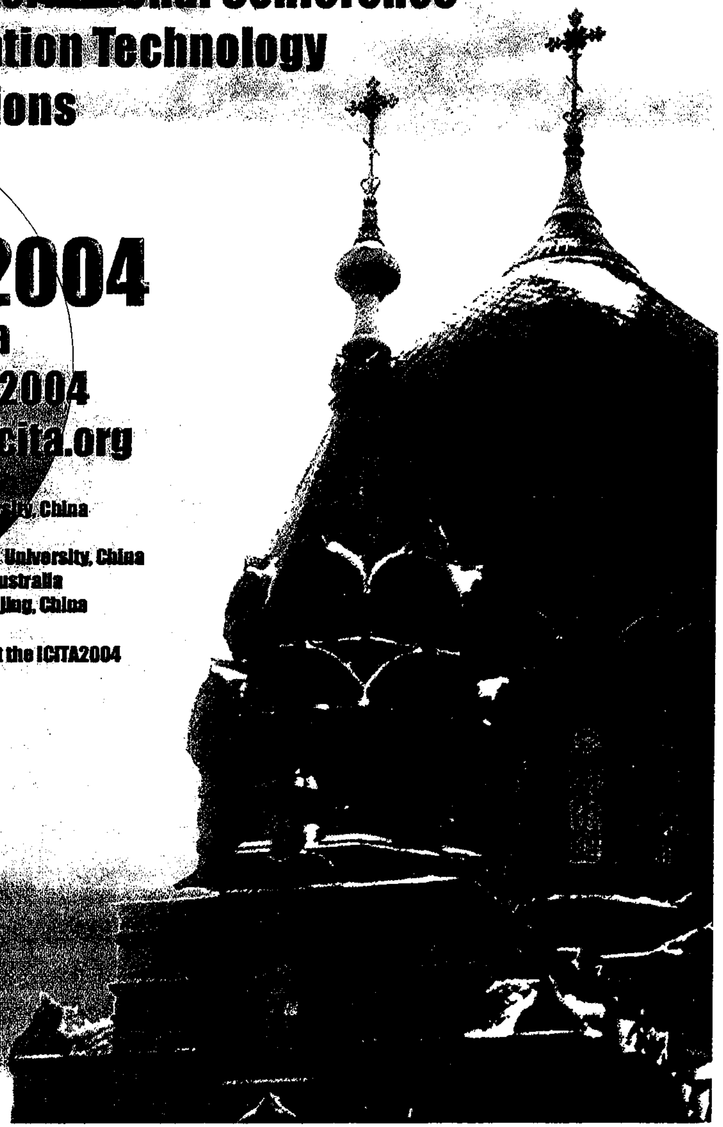
Harbin, China

8-11 January 2004

http://www.icita.org

[illegible]
[illegible], China
[illegible]
[illegible] University, China
[illegible] Session, Australia
[illegible] CS Chapter, Beijing, China
Saora Inc., Japan
papers published at the ICITA2004 will be EI indexed.

IEEE



Contents

Theme 1: IT in Telecommunication and Mobile Communications

Theme 2: IT in Multimedia; Computer Networking; and Database Interface

Theme 3: IT in Image Compression.

Theme 4: Web Content Generation, Usage and Management.

Theme 5: IT in Engineering: AI, Signal/Image Processing; Power & Power Electronics; Sensors.

Theme 6: IT in Education and Open Learning.

Theme 7: IT in Health, Medical Care and Biomedical Engineering.

Theme 8: Security and Hacking.

Theme 9: E-commerce.

Theme 10: IT in Animation and Graphic Design.

Theme 11: IT in the Humanities and Social Science.