A Multi-Agents System for Program Mining: Architecture, Interaction Protocol and Implementation

Fang Cunhao,† and Zhang Yaoxue††,

Tsinghua University, Beijing, China

Summary
With the rapid development of computer network, Internet is used not only to publish and share Information, but also to provide many kinds of computing services. Program Mining is presented to deal with customized computing problem on demand for Internet users. This paper first introduce the basic concept of Program Mining: Program Mining makes use of several task-specific software agents, analyzes user requests for computing, searches the component candidates from online component libraries according to the request, and reassembles them to form programs that perform the expected computing.According to the general process of Program Mining, a multi-agent system for Program Mining is designed and the communication protocols between the agents are also developed. After analyzing the function of each entity in Program Mining system, we design one or more agent to act as the each entity. Then for each agent, the function and the process are provided in detail. Based on the Agent Communication Language, the communication protocols are presented for the multi-agent system. Via the communication protocols, it is assured that the actions of agents are consistent to the whole multi-agent system. In the implementation of multi agent system, we have developed a agent interaction protocol and language based dynamic agent infrastructure, which applies XML to specify the messages among mobile agents and define their tasks and access rights, to support multi-agent cooperation for program mining.Input here the part of summary.

Key words:
Multi-agent, Interaction Protocol, Program Mining, Computing-on-demand

Introduction

For the past few years, the Internet is often regarded as a super information database, where information related to a wide range of topics is disseminated and shared. To help users find needed information on the Internet, many efforts have been made to develop information on demand systems, such as news on demand, video on demand, or other personalized or intelligent information retrieval systems. Some researchers implement Data Mining techniques to discover knowledge from Web sites, we regard it as knowledge on demand approach [1,4].

However, with the evolution of network technology, especially the prosperity of Java, the Internet is emerging as a large-scale distributed computing platform, where all kinds of Internet applications (web services) are deployed, providing various services for users. Traditionally, these applications often employ prepackaged monolith systems containing any conceivable features, which are not easily extended and customized. Whereas in large scale distributed networks (e.g., the Internet), network services and applications are diffused to a very large scope. This makes it necessary to increase the customizability of services, so that different classes of users in heterogeneous networks can tailor the functionality and interface of a service according to their specific needs [1]. An ideal solution to this problem is to implement applications as component-based systems and deliver them at an on-demand manner, so that new features can be added on demand at different granularities. Therefore, in Internet world, besides the need for personalized information, users have similar needs for computing. They need customized computing functionalities to process customized information. We regard it as the need for computing-on-demand at application level.

Based on these observations, we propose a new computing paradigm—Program Mining (PM) to deal with the increasing needs of computing-on-demand. The basic idea of PM is making use of several task-specific software agents, analyzing user's requests for computing, searching and retrieving candidates from online component repositories according to this request, composing and reassembling them to form programs that perform the expected computing. In this way, computing on demand can be provided for users, achieving great flexibility and customizability.

The rest of this paper organize as follows: In section 2, we present a more concrete concept of Program Mining; Based on this concept, we discusses and a multi-agent system framework for program mining. In section 3, we

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.
This paper is sponsored by National Natural Science Foundation of China (NO. 90604027)

present the agent interaction mechanism for component retrieval with XML messaging. Finally, we summarize this paper by drawing a conclusion.

2. The Concept Diagram of Program Mining

In our proposal, software agents are utilized to discover programs among large amount of component resources. The users present what they want in terms of functionalities, which may be described in natural language and the concept of component is transparent to end users. The request is analyzed and the computing functionality is decomposed into smaller modules, possible components or component compositions that can realize these functional modules are identified. Then corresponding software agents are activated to search and retrieve potential component candidates, analyze and discover the dependencies among them, find out the possible component compositions that implement the needed computing logic. Just as Data Mining systems that discover implicit relations and patterns in a vast amount of data, Program Mining is to discover the dependencies and relationships among a great deal of software components, and compose executable programs using various task-oriented software agents. In the intermediary nodes of active networks, this mechanism can be used to dynamically discover programs that provide needed active services; in end systems, some network applications can also be composed on-the-fly using ProgramMining. The concept of Program Mining is depicted in Fig.1.

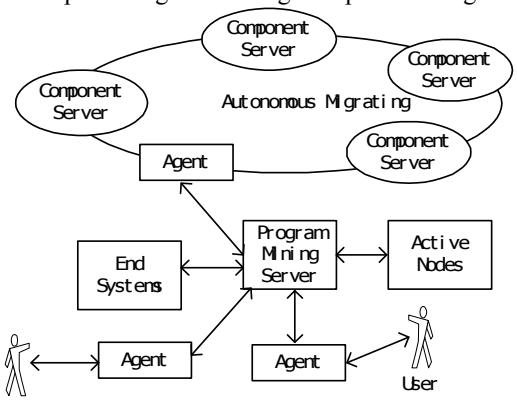

Fig. 1. Illustrates the basic concepts of Program Mining.

We suppose that reusable software components have been organized in different component servers on the network. Software agents migrate autonomously among these servers to search and retrieve potential component candidates. On the Program Mining server, according to users' high-level requests, application level programs and network level computing logic can be discovered from the candidates. Particularly, in active nodes, active services are provided for the end users to perform customized computation, forming the programmable network API of active nodes. Software agents on the Program Mining server can carry out the composition of these services. In end systems, applications that utilize the underlying network services can also be composed using Program Mining mechanisms. Depending on the complexity of mining tasks, user intervention with agents might be needed when necessary.

3. The Multi Agent System Architecture for Program Mining

To provide users with Active Services, the program mining system needs to access and search the heterogeneous component warehouses distributed on the Internet and LANs, as well as realize the co-sharing and reuse of the component resources distributed in these component warehouses. In the process of program mining, it is necessary to make functional decomposition of user service requests, and make a judgment upon whether the service offered by the component can meet functional needs. If the services offered by the components can meet the demand of the user, the system can directly return the service functionality offered by the component to the user. Otherwise, the system will further the component searching and matching according to the service functions after further subdividing. After the relevant components are found, the system will compose, compile, validate, and test them. Finally, the system will submit the requested services to the user. Besides, the intelligent agent can also record the former cases and and learn them to increase its capability, no matter whether they are successful or not.

3.1 Agents Involved in Program Mining Process

In Section 2, we have introduced the general process of program mining:
(1)the user submitting computing requests;
(2)the system analyzing user requirements and making a functional decomposition;
(3)component searching and acquiring;
(4)component analyzing, selecting, and composing;
(5)validating the consistency of the composing program and the service required;
(6)compiling and executing the mined program and offering services to the user.

To implement the program mining progress, the program mining system should set up relevant intelligent agents in the client and server, and organize them in accordance with a certain protocol to constitute a multi-

agent system in support of program mining. We propose and implement a multi-agent cooperation model for program mining. This multi-agent system consists of the following agents:

- **User Interface Agent** (UIAgent) at the client side
- **Task Management Agent** (TMAgent) at the program mining server side
- **Task Analysis Agent** (TAAgent) at the program mining server side
- **Component Retrieval Agent** (CRAgent) at the program mining server side
- **Composing and Validating Agent** (CVAgent) at the program mining server side
- **Domain Knowledge Agent** (DKAgent) at the domain knowledge base side
- **Directory Library Agent** (DLAgent) at the component warehouse side
- **Component Library Agent** (CLAgent) at the component warehouse side.

Now, we briefly discuss the main functions of these agents.

- ***UIAgent.*** The UIAgent (User interface agent) is located at the client end. It interacts with, guides, and helps the user to put in program mining task, or in other words, accepts the user's computing requests actively. Then, it submits the computing requests to the program mining server and records the user's requests using history and characteristics. After program mining servers return mined components or component sets, it will provide the user with these components.

- ***TMAgent.*** The TMAgent (Task management agent) receives the computing requests from the UIAgent, and allocates the needed resources to these requests. Meanwhile, the TMAgent is responsible for recording the status of each task, including the execution status and result of the task. Here,we term every single computing request from the user as a task.

- ***Task Analysis Agent (TAAgent).*** The task analysis agent is responsible for user requirement analysis and corresponding function decomposition. We term functions that cannot be further subdivided as primitive functions, which corresponds to a service function provided by a single component. Here, we should note that, due to the heterogeneity of components and the diversity of developers, it is inevitable to witness a component whose functions contain another component's functions, or the functional intersection with another component. It thus demands that the TAAgent should be equipped with the capability of partitioning primitive functions and recording the partition process and history. Before decomposing of primitive functions, the task analysis agent demands that the domain knowledge agent supply the classification information about the relevant components in the component warehouse, as well as the component's service functions. After the task analysis agent decomposes the user requirements into a group of primitive functions, it will transfer the primitive function requests to the component retrieval agent.

- ***Component Retrieval Agent (CRAgent).*** The component retrieval agent is a movable agent, which is used to search and acquire the components that can meet the required primitive functions from the distributed component warehouses or local ones. It has two functions: (1)Accept requests from the task analysis agent and acquire the component resource information from the component directory library, or, according to the component resource information, move to the corresponding component warehouse, and query and acquire the component entity. (2)Accept the request of the component directory library agent, search in the stored component information in each component warehouse, and update the component resource information in the component directory library.

- ***Composing and Validating Agent (CVAgent)***. When the component retrieval agent has found the needed component, it transfers the component's information to the CVAgent. The CVAgent, according to the user requirements and the Component Composing scheme, composes the components into an application program that can offer the service required by the user, and then validates its consistency. Finally, the CVAgent compiles and runs the composed and validated application program to offers active service for the user.

- ***Domain Knowledge Agent (DKAgent).*** The DKAgent accepts the request from the TAAgent and sends back the relevant domain knowledge and component service function to the TAAgent after searching the domain knowledge base. The domain knowledge includes the domain classification information and the function sets information in the domain. In addition, the domain knowledge agent is responsible for updating the domain knowledge base.

- ***Directory Library Agent (DLAgent).*** The DLAgent accepts the request from the CRAgent, and, after querying the component directory library, sends back the relevant component resource location information to the CRAgent for the component searching. Besides, it also accepts the component resource's update

information obtained from the search engine agent, and maintains and updates the component's resource information.

- ***Component Library Agent (CLAgent).*** The CLAgent accepts the request from the component retrieval agent, and, after querying the component warehouse, sends the relevant component information and component entity to the component retrieval agent. Meanwhile, the CLAgent is also responsible for the management and maintenance of the components in the warehouse, including the addition, deletion, or modification of the components.

3.2 Multi-Agent Cooperation Model

The collaborative work relationship among the above agents in the mining system are shown in Figure 2.

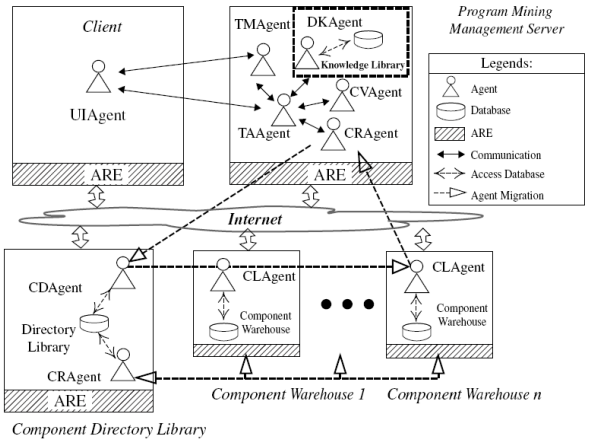

Fig. 2 Cooperation Model of multi-agent system in program mining

In Figure 2, the user interface agent is set at the client to complete interactive functions with the user. Other agents are located in the component management server, component directory library, component warehouse, etc. Since the intelligent multi-agent system is an Internet-based distributed computing system, the locations of these intelligent agents in the above-mentioned servers, directory library, or component warehouse have relative and changing positions.

The communication between different agents is directed by the protocol connecting them.

Figure 3 indicates the protocol link relations among these intelligent agents. As shown in Figure 3, the connective protocol between the UIAgent and TMAgent is UITM. Moreover, the connective protocol between the TMAgent and task analysis agent is TMTA, the UIAgent and task analysis agent, UITA, the task analysis agent and domain knowledge agent, TADK, the task analysis agent and component retrieval agent, TACR, the task analysis agent and composing and validating agent, TACV, the component retrieval agent and directory library agent, CRDL, and the component retrieval agent and component warehouse agent, CRCL.

These protocols define the information interactive rules, formats and orders among intelligent agents. The interactive messages are described by the relevant agent communication languages.

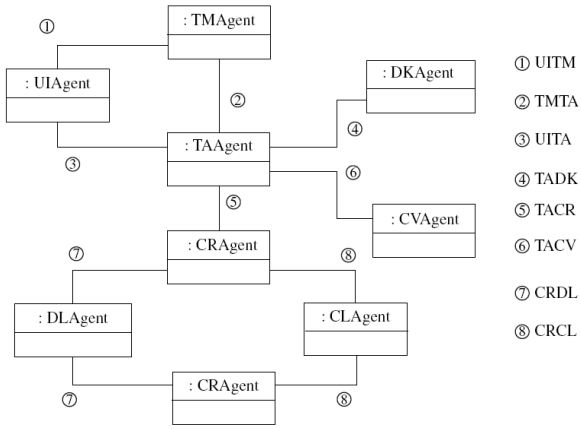

Fig. 3 Protocols of intelligent agents.

4. Agent Interaction Protocol and Language

4.1 Agent Interaction Protocol

This section is concerned with the agent's interaction protocol. Because the agent interaction protocols between intelligent agents depends on the task classification of each agent and functional interrelationship between agents, and also because there is little difference between interaction protocols themselves, we only specify UITM the interaction protocol between the UIAgent and TMAgent.

Figure 9.5 is the state transition of a UIAgent in UITM. Figure 9.6 is the state transition of the TMAgent in UITM. In the figures, the state of the agent is indicated by an ellipse. The arrow refers to the target directions of the state transition. The notes above the arrow with "+" indicate the received message during the state transition, and indicate the sent message with "-" below the line.

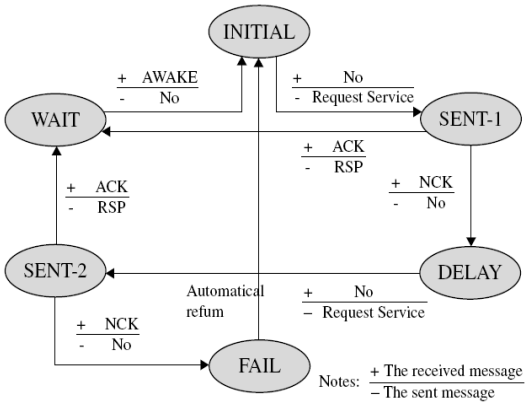

Fig. 4 State transition map of the UIAgent in UITM.

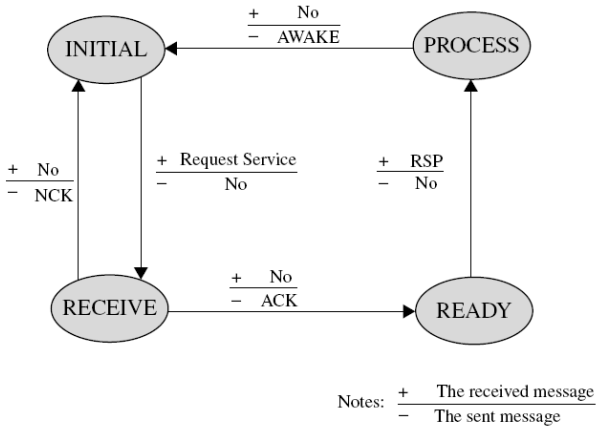

Fig. 5 State transition map of the TMAgent in UITM.

As shown in Figure 9.5, in the whole interaction process of the UTIM protocol, the UIAgent should undergo six states: INITIAL, WAIT, SENT-1, SENT-2, DELAY, and FAIL. The UIAgent's state transition is triggered by sending or receiving of the message. Initially, the UIAgent is in state INITIAL. When the user inputs the request, the UIAgent packs it into the message "Request Service"and sends it to the TMAgent, meanwhile shifting the current state to SENT-1. If the TMAgent is busy, it answers "NCK". Then, the UIAgent changes to state DELAY, and resends the message "Request Service" after a while, shifting to state SENT-2. If TMAgent replies "NCK" for two successive times, the UIAgent will shift to state FAIL, reporting to the user that this task fails and then returning to state INITIAL automatically. If the UIAgent, in state SENT-1 or SENT-2, receives "ACK" from the TMAgent, it means that the TMAgent has accepted "Request Service." In turn, the UIAgent will send "RSP" to the TMAgent, requesting the TMAgent to process "Request Service" and shifting to state WAIT. When the program mining server returns mined components or component sets, it is up to the TMAgent to send "AWAKE" to UIAgent, which is then awakened and shifts its status to state INITIAL, submits the mining results to the user, and starts to accept the next request service.

Figure 5 is a state transition of the TMAgent in the UITM protocol. In the beginning the TMAgent is in state INITIAL. When receiving the "Request Service"message from the UIAgent, TMAgent will decide whether to accept this request of service or not, according to its service type, current resource status, etc. If it has been declined, the TMAgent will answer a "NCK" and return to state INITIAL, waiting for the next service request. If accepted, the TMAgent will answer "ACK," shifting to state READY. After receiving the UIAgent's message "RSP," the TMAgent will go into state "PROCESS" and work with other agents to jointly process the service request of the "Request Service" message. When it is done, the TMAgent sends a "AWAKE" to the UIAgent to awaken it for corresponding processing. Then, the TMAgent returns to state INITIAL.

4.2 Agent Interaction Language

The agent interaction protocol prescribes the rules, sequence, and formats of agent communication. The agent interaction language is used to describe the messages among agents. Different from the messages transmitted by other protocols, messages of the intelligent agents contain three basic elements: the agent interaction language, the universal format of message content, and the ontology defined among the agents for mutual understanding.

XML can be used to describe the messages exchanged between agents. An example of a message format defined by XML is as follows:

Example 1 The common format of message described by XML

```
<Message>             <!-- Message Starts -->
   <protocol> </protocol> <!-- Communication Protocol -->
   <type> </type>         <!-- Type -->
   <sender> </sender>     <!-- Sender-->
   <receiver> </receiver> <!-- Receiver-->
   <ontology> </ontology> <!--The common terms understood
                                  by multi-agents-->
   <content>  </content>  <!-- Content -->
   <language> </language> <!-- Encoding Language-->
</Message>             <!— Message Ends—>
```

Each message has a sender, and one or more receivers. The "content" in a message corresponds to the concrete content sent in the message.

The "ontology" in a message refers to the commonly understood terms by the intercommunicating agents. It is a description of some terms contained in the content, such as objects, methods, etc. For example <content> <Domain name="car">...</Domain></content>contains a domain object named car. Domain is an ontology understood by multi-agents. The definition is given as follows:

Example 2 Domain's definition in ontology

```
<!ELEMENT Domain (FeatureList, FeatureTree)>
   <!-- Domain contains two elements, namely, FeatureList,
                                          FeatureTree -->
<!ATTLIST Dormain name CDATA #REQUIRED>
   <!--  Define the name of Domain -->
<!ELEMENT FeatureList (Feature*)>
   <!-- FeatureList lists all features of this domain-->
<!ELEMENT Feature (name, description, source, category,
  variability, bindingTime, issueAndDecision, notes,
  include, specializedBy, implementedBy, require, mutex)>
   <!-- Features' definition-->
<!ATTLIST Feature FeatureID ID #REQUIRED>
  <!-- Assign a unique label, FeatureID, to each feature-->
<!ELEMENT FeatureTree (FeatureTree*)>
  <!--FeatureTree's recursion definition. It is made of
                                    FeatureTree-->
<!ATTLIST FeatureTree FeatureID IDREF #REQUIRED>
  <!--Each node of FeatureTree is FeatureID, which corre-
sponds to a feature of this domain defined in FeatureList -->
```

Example 3 The XML description of the message " Request Service" to the TMAgent, sent by the UIAgent

```
<?xml version="1.0">
<message>
   <protocol> TCP </protocol >
     <!--it means that this message is transferred
       through TCP protocol-->
   <type>Request Service</type>
     <!--Message type is Request Service-->
   <sender link=http://agent.tsinghua.edu.cn/-UIAgent>
     UIAgent
   </sender>
     <!--The sender of message is UIAgent-->
   <receiver link=http://bean.tsinghua.edu.cn/~TMAgent>
     TMAagent
   </receiver>
     <!--The receiver of message is TMAgent-->
   <ontology link=http://agent.tsinghua.edu.cn/~ontology>
     File Format Viewer
   </ontology>
     <!--The ontology of message is"File Format Viewer"-->
   <content>
     <!--Request for a Java-based component for
       FTP applicaiton-->
     <application name= a network application>
       <language>java</language>
       <function>protocol</function>
     </application>
   </content>
   <language link=http://agent.tsinghua.edu.cn/kif.html>
     ACML
   </language>
</message>
```

5 Agent Function Design and Implementation

In this section, we give an introduction to the function designs of agents, because the functions each intelligent agent is about to fulfill depend on the system designer's allocation of each agent's tasks, and on the functionality and supporting environments of the selected intelligent agent platform. The functional design of the UIAgent is illustrated as an example.

Eample 4 The function design of the UIAgent

As stated, UIAgents are mainly used to help and guide the user to accomplish program mining. The UIAgent, by means of learning, will adapt to the user's preference, and automatically run some commonly used procedures. Generally speaking, the UIAgent adopts four approaches to learning:

(1)observing the user's operations and conducting imitation learning;

(2)making suggestions to the user, or executing operations on behalf of the users, then learning and adjusting itself through receiving the feedback or evaluation from the user;

(3)directly accepting the user's commands and then learning and recording relevant operating flows;

(4)consulting other agents for their experience.
With such a broad learning mechanism, the interface agent can offer users personalized interaction interfaces to customize users' own interaction styles. Besides, interface agents can co-share the knowledge by bilateral collaboration. When a user customizes or modifies an application service with the help of the program mining system, the UIAgent can learn and remember this procedure and lay a foundation for other users to share the service in future.
Figure 6 illustrates the module structure of the UIAgent, which consists of the sensor module, reasoning machine, communication module, knowledge base, database, controller, and effect module. Users submit computing service requests through the input/output interface. When the sensor module receives the user's computing service requests, it will send messages to the reasoning machine, which analyzes the sensed events according to the rules in the knowledge base and the statistical data in the database. Then, the reasoning machine will decide the subsequent actions in accordance with the analysis result. Particularly, if an event has been input by a user, the application result of this event will serve as a reference for future handling of the user's input. Similarly, when the same user has been using the agent for a while, the UIAgent is supposed to memorize the user's input habit and character, such as the frequently used input modes or frequently requested services.
Therefore, once the sensor module perceives that one user's input can match the related habit or character, the reasoning machine can instantly confirm the user requested services, and start corresponding, subsequent acts. The acts are sent to the controller, which will trigger them in some arranged sequence. The controller executes the acts according to their concrete contents, which ranges from sending messages to other agents, generating new rules and putting them into the knowledge base, recording the latest statistical data, or outputting information through the effect module.
The UIAgent can intercommunicate with other agents through communication modules. As shown in Figure 3, the communication objects of the UIAgent are the TMAgent and TAAgent, and the communication protocol and message have be elaborated in Section 4.

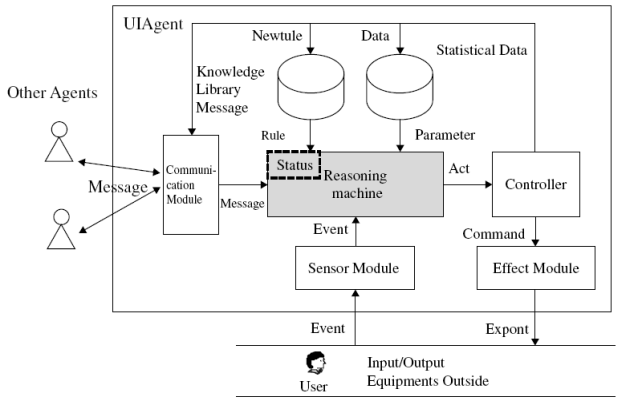

Fig 6 Functional structure of the UIAgent.

6 Conclusions

With the rapid development of computer networks and communication infrastructure, the Internet accessing technologies and devices are becoming more diverse. End users want to customize the functional feature sets of programs according to the network environment and resource constraints. Aiming at this problem, we propose a new computing paradigm— Program Mining to approach computing on demand in distributed environments.
We also discussed the basic concepts, a multi-agent system framework for Program mining general process in program mining. With the penetration of Internet into everyone's daily life, we believe that the change from information/knowledge-on-demand to computing-on-demand will be an important trend in the way people using Internet. More efforts should be made to achieve this goal. Program Mining is an initial attempt to approach it. Although the work in this paper represents only a beginning, we feel that Program Mining as a new computing paradigm offers a broad new field of research.

Acknowledgement

We would like to thank the support of the National Nature Science Foundation of China (No. 90604027).

References

[1] Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, Vol. 24, No. 5, May 1998.

[2] D.Tennenhouse and D.Wetherall, "Towards an Active Network Architecture", Computer Communication Review, 26(2), April 1996.
[3] L.A.Guedes, P.C. Oliveira, L.F. Faina and E.Cardozo, "An Agent-based Approach for Supporting Quality of Service in Distributed Multimedia Systems". Computer Communications 21 (1998) 1269-1278.
[4] Minos N. Garofalakis, Rajeev Rastogi, S.Seshadri and Kyuseok Shim, "Data Mining and the Web: Past, Present and Future" WIDM99 Kansas City Mo USA.
[5] RIG Uniform Data Model for Reuse Libraries (UDM), RPS-0002, Reuse Library Interoperability Group, January 1994.
[6] Sanjiva Weerawarana and Matthew J. Duftler, "Bean Markup Language (Version 2.3) User's Guide", http://www.alphaWorks.ibm.com/formula/bml.
[7] The Common Object Request Broker: Architecture and Specification, Version 3.0, CCM FTF Draft ptc/99-10-04, 29 October 1999.
[8] Wienberg A, Matthes F and Boger M, "Modeling Dynamic Software Components in UML", UML'99, 1723: 204-219, 1999.
[9] XMI http://www.software.ibm.com/ad/features/xmi.html.
[10] Fang, C. H., Zhang, Y. X., and Xu, K. G. (2003). An XML-based data communication solution for program mining. Intelligent Data Engineering and Automated Learning, 4th International Conference (pp. 569–575). Hong Kong, Berlin Heidelberg: Springer-Verlag.
[11] Hull, R., Benedikt, M., Christophides, V., and Su, J. W. (2003). E-Services: a look behind the curtain. Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 22, 1–14.
[12] McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. IEEE Intelligent Systems and Their Applications, 16(2), 46–53.
[13] Nunez, S. J., O'Sullivan, D., Brouchoud, H., et al. (2000). Experiences in the use of FIPA agent technologies for the development of a personal travel application. Proc. of the International Conference on Autonomous Agents (pp. 357–364).
[14] O'Sullivan, D., and Lewis, D. (2003). Semantically driven service interoperability for pervasive computing. Proc. of the Third ACM International Workshop on Data Engineering for Wireless and Mobile Access: MobiDE 2003 (pp. 17–24).
[15] Raman, B., and Katz, R. H. (2003). Load balancing and stability issues in algorithms for service composition. Proceedings—IEEE INFOCOM,2 (pp. 1471–1487).
[16] Tosic, V., Pagurek, B., Esfandiari, B., et al. (2002). Management of compositions of E- and M-business web services with multiple classes of service. IEEE Symposium Record on Network Operations and Management Symposium (pp. 935–939).
[17] Vidal, J. M., Buhler, P., and Stahl, C. (2004). Multiagent systems with workflows. IEEE Internet Computing, 8(1), 76–82.
[18] Xia, D. L., Zhang, Y. X., and Fang, C. H. (2003). Design and implementation of an agent-based program mining system. Chinese Journal of Electronics, 31(5), 793–796.