An Agent-Based Collaborative Architecture for Knowledge-driven Process Management

Aizhong Lin, Brian Henderson-Sellers, Igor Hawryszkiewycz

Faculty of Information Technology
University of Technology, Sydney
POBox 123, Broadway
NSW 2007, AUSTRALIA
{alin, brian, igorh}@it.uts.edu.au

Abstract

This paper proposes an agent-based collaborative architecture for collaborative work. This architecture is applied to knowledge-driven process management, which often requires quick reconfiguration of applications. Using this architecture, an open multi-agent system is integrated with a virtual collaborative environment. The virtual collaborative environment provides the representation and management for knowledge-driven processes. The open multi-agent system is quickly reconfigurable to provide suitable autonomous functions that perform the knowledge-driven processes.

1 Introduction

A *business process* is a set of related activities to achieve a business goal. Debenham [3] categorizes business processes as activity-driven processes, goal-driven processes, and knowledge-driven processes in terms of their manageable properties. A *knowledge-driven process* is a specific business process whose goal or activities may not be completely specified in advance but emerge over time as knowledge is gained from the activities performed earlier. In other words, it is the growing body of knowledge that provides the direction to knowledge-driven processes. In its decomposition, a knowledge-driven process contains goal-driven processes or activity-driven processes as its sub processes. Knowledge-driven processes are normally collaborative because the knowledge comes from not only the single process participant's work but also the process participants working in a team. Managing knowledge-driven processes aims to provide knowledge representation and management functions that support collaborative work so as to respond to progress in executing the process more quickly, to reduce process cost, and to increase the process performance.

This research provides an agent-based collaborative architecture for knowledge-driven process management. The architecture is an integration of an open multi-agent system (OMAS) with a virtual collaborative environment (VCE). The VCE in this architecture represents and manages knowledge for knowledge-driven processes and the OMAS provides reconfigurable autonomous process functions to support the achievement of process goals, performance of process activities, and rapid response to progress of the process enactment.

A business process in a virtual collaborative environment is decomposed into related process elements that are goals, activities, roles, participants and artifacts. A workspace model is defined in the virtual collaborative environment to represent and maintain these process elements. The workspace model is particularly suitable for representing the growing body of knowledge that directs knowledge-driven processes because the growing body of knowledge can be represented as a workspace tree [6].

The multi-agent system is open due to its reconfigurable property. An intelligent agent can freely join the multi-agent system or quit from the multi-agent system and one group of agents may provide different functions from another group of agents. Furthermore, the multi-agent system can provide autonomous functions because an *intelligent agent* as defined here is a software component, situated in some environment, that is capable of autonomous and flexible actions to respond to changes in the environment in order to achieve process goals [9].

The paper describes the agent-based collaborative architecture. Firstly, the collaborative architecture is introduced using a high level model which illustrates how the open multi-agent system is integrated with the virtual collaborative environment. Secondly, the ontology of the virtual collaborative environment and the open multi-agent system are outlined. In particular, a reusable intelligent agent architecture and the multi-agent interaction are described. Then, the collaborative architecture that contains the open multi-agent component and the virtual collaborative component is illustrated in detail. Finally, a knowledge-driven process management application built on this architecture is described. This application is currently utilized in a university environment to support knowledge intensive works.

2 The Agent-based Collaborative Architecture

The agent-based collaborative architecture is illustrated in Figure 1. Users or participants work together in a virtual collaborative environment (as described further in section 3). Any change taking place in the collaborative environment made by one or more users is also represented as an event in the environment. Events can be detected by agents in the open multi-agent system, as described in section 4. Only the responsible agents employ the events to drive actions. The actions, after they are executed, may change the virtual collaborative environment and those changes produce new events.

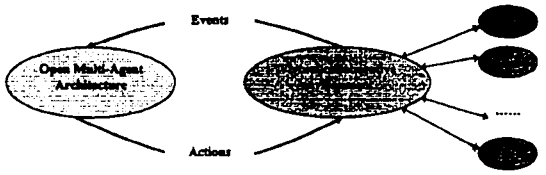

Fig. 1. A high level model of the agent-based collaborative architecture

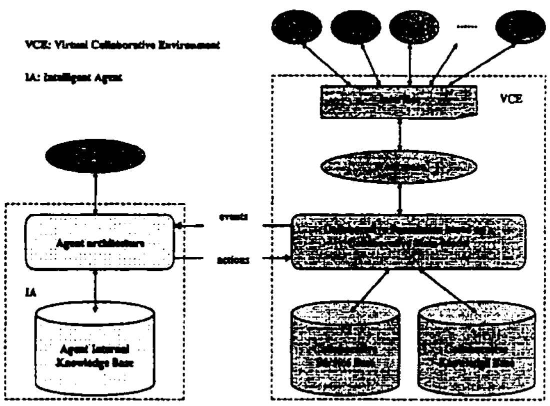

Fig. 2. The description of an intelligent agent connects to the virtual collaborative environment

Fig. 2 illustrates how a single agent in the open multi-agent system connects to the virtual collaborative environment. There is a collaborative database that contains a description of the current collaborative activities. These activities can use any of the collaborative services provided by the collaborative environment. The open multi-agent architecture contains the agents and their associated knowledge bases. These are linked to the collaborative environment through "events" (from the virtual collaborative environment to an agent) and "actions" (from an agent to the virtual collaborative environment). Collaborative services (e.g. create_a_new_workspace) built by users can be integrated into an agent as the actions of the agent and those actions stored in the agent internal knowledge base.

3 The Virtual Collaborative Environment (VCE)

The virtual collaborative environment is based upon a collaborative meta-model adapted from [1] (Fig. 3). The meta-model centres on activities that could be made up of a number of sub-activities as indicated by the looping arrow. An activity, and its sub activities, is represented in a workspace. A person, here called a participant, 'is-in' a group and occupies one or more roles. A group, which can evolve independently, may contain subgroups. An activity 'has' any number of roles and events and 'contains' any number of views, which contains artifacts or define groups of artifacts. The roles define access rights to views and can 'take' actions. Activities can access workitems, which are composed of a number of actions. Actions 'use or create' artifacts. An action could be a soloaction, which is taken by an individual, or an interaction, such as a discussion, which includes more than one participant. An activity can cause a number of events, which are used to 'drive' workflows. A workflow 'is in' an activity and is composed of a sequence of workitems.

The major elements of the collaborative environment are Workspace, Activity, Group, Role, Participant, View, Artifact, Workitem, Action, Interaction, Workflow and Event. They are specifically defined as follows.

- *Workspace* — a workspace is an interface that supports the representation of an activity including its sub activities
- *Activity* — an activity maintains a collection of other process elements. For example, it has roles and events, contains views, and can access workitems. The workitems are composed of actions that can produce defined outputs
- *Group* — a group is a collection of participants
- *Role* — a role defines a set of responsibilities in an activity or a sub activity

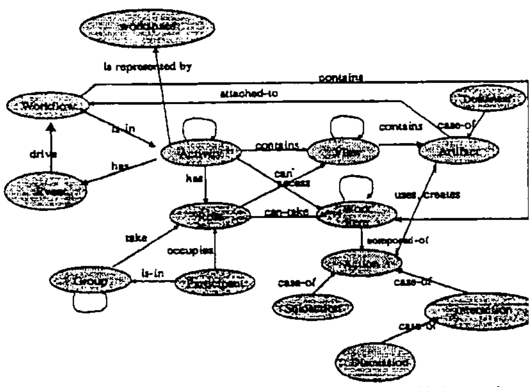

Fig. 3. The meta-model of the virtual collaborative environment (adapted from [1])

- *Participant* a participant is a specific person assigned to a role
- *View* a view is a collection of artifacts
- *Artifact* an artifact is an information object such as a document
- *Workitem* a workitem is a set of actions needed to produce intermediate outcomes that eventually produce an output. A workitem is composed of a number of actions.
- *Action* an action is a specific unit of work carried out by a role (e.g. Change an artifact, send an artifact)
- *Soloaction* a soloaction is an action that is executed by an individual participant
- *Interaction* an interaction is a set of message exchanges between people when they collaborate in activities. An interaction may not produce an explicit output although it may change people's knowledge
- *Workflow* a workflow is a sequence of workitems that are driven by events
- *Event* an event is when something takes place in an activity or a sub activity. For example, a document is replaced.

The virtual collaborative environment has been implemented using the Java 3-tier technology [7]. The interface of the workspace (shown in Fig. 2) is implemented using Java Server Page (JSP) and Java Servlet, the collaborative foundation (shown in Fig. 2) is implemented using Enterprise Java Bean (EJB), and the collaborative knowledge base and the collaborative service base (shown in Fig. 2) is implemented on the Sybase database system.

4 The Open Multi-Agent System (OMAS)

The open multi-agent system (OMAS) takes events from the VCE as its input parameters and provides actions to the VCE as output in order to achieve collaborative goals defined in the VCE. To describe the open multi-agent system, we introduce the open multi-agent architecture, the multi-agent interaction protocol, the intelligent agent architecture, and the agent internal knowledge base. The open multi-agent architecture describes how multiple agents are organized, the multi-agent interaction protocol illustrates how agents interact with each other, the intelligent agent architecture describes how an individual agent achieve its goal and the agent internal knowledge base describes the agent internal knowledge and how it is maintained.

4.1 The Open Multi-Agent Architecture

The open multi-agent system is a collection of individual intelligent agents that work together for a common goal. An agent in the open multi-agent system can join the system or leave the system freely. In the virtual collaborative environment, an agent is taken as a participant. Different agents have different actions to achieve different goals. They are organized in the multi-agent system to interact with each other to achieve collaborative goals.

The architecture to support the open multi-agent system is illustrated in Fig. 4. When an agent joins this system, it sends a "register" message that contains its personal information (e.g. ID, name, creator, birthdate and goals) to the PIMA (personal information management agent) situated in the VCE. The PIMA saves the agent personal information in the agent personal information database. Similarly, before an agent leaves the multi-agent system, it sends a "unregister" message to the PIMA so that the PIMA deletes the personal information from the agent personal information database. The open multi-agent system can be dynamically organized because agents can dynamically join or leave the system.

To achieve a common goal, agents interact with other agents to share actions or knowledge. Agents interact with each other by exchanging messages. A message carries a piece of information or knowledge that the sender wants the receiver(s) to know. In the open multi-agent architecture as shown in Fig. 5, an MMA (message management agent) situated in the VCE receives messages from agents and then dispense them according to the destination agents. An agent message has a specific format that either obeys KQML (Knowledge Query and Manipulate

Language) [4] or the ACL (Agent Communication Language) [5] standard. In the open multi-agent system, agents use ACL to express messages.

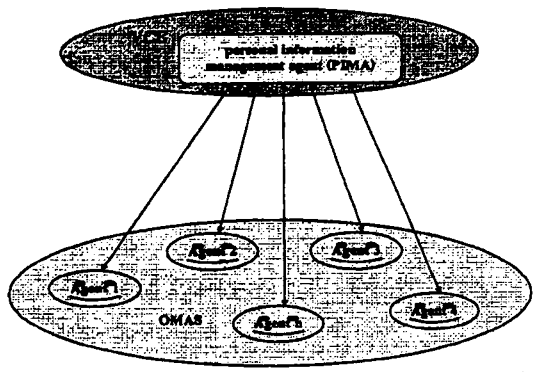

Fig. 4. The PIMA supports agents joining and leaving the OMAS dynamically (adapted from [8])

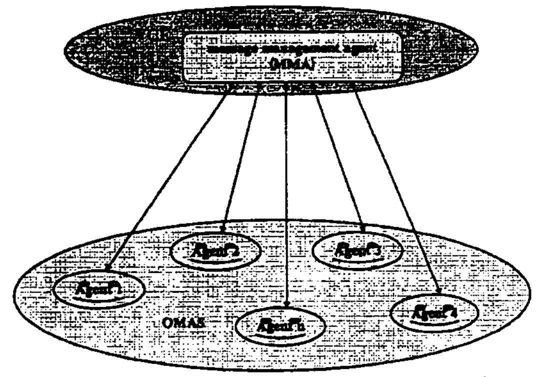

Fig. 5: The MMA supports agents exchanging messages between agents

4.2 Multi-agent Interaction

Agents interact with each other by exchanging messages. The interaction messages are normally constrained by interaction protocols. An interaction protocol defines a set of interaction messages that can be understood by the agents. Fig. 6 illustrates an example of a "delegation" interaction protocol by which one agent asks another agent to achieve a goal. From the beginning, the interaction initiator (requester) sends a message that carries the "delegate" information to another agent (receiver) The receiver agent could "not-understand", "refuse" or "commit" the request. Once a commitment is made, the actions assocaited with this commitment are taken at the scheduled time. However, the result of doing those actions could be cancel, failure or success. If it is failure, the reason for failure is sent to the requester. If it is successful, the success with the result is informed to the requester. Finally, if it is cancelled, the reasons for this are provided. After the requester agent checks the results, it decides to terminate the interaction. Two interaction protocols are designed in the OMAS. One is for the "delegate" and the other is for a negotiation to "share" a piece of knowledge.

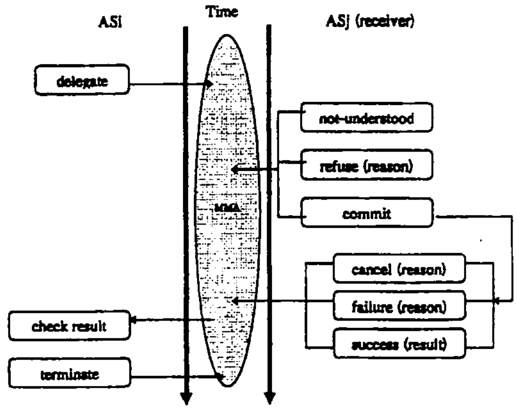

Fig. 6: An interaction protocol for "delegation" by which one agent asks another agent to achieve a goal. The interaction messages are managed by the MMA

The MMA is built to manage all messages exchanged between the two parts of the "delegation" interaction. The primitives used in this protocol are:

- *delegate*: A "delegate" primitive initializes an interaction in which the sender asks receiver to do something. For example, one agent requests another agent to find a subject instructor
- *not-understand*: The receiver agent does not understand the semantics of the request
- *refuse*: The receiver agent refuses to do the task for the sender for some reasons such as no time
- *commit*: The receiver agent commits to do the task in a specified time
- *cancel*: The commitment is decommited for some reason
- *failure*: The commitment is failure when it is performed
- *success*: The commitment is done and the results is placed in the agent interaction workspace
- *terminate*: The interaction initiator decides to terminate the interaction after the results have been checked

4.3 The Reusable Intelligent Agent Architecture

Individual intelligent agents in the open multi-agent architecture have a reusable agent architecture defined by the concepts and relationships between the concepts as shown in Fig. 7.

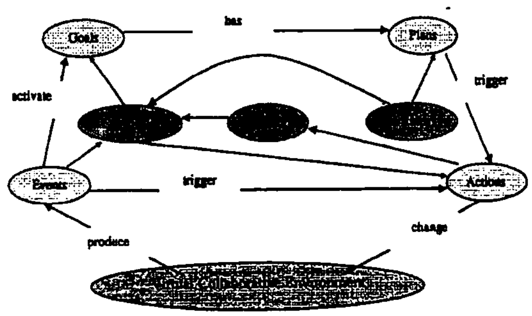

Fig. 7. The reusable intelligent agent architecture

Events produced in the VCE can be detected by an agent. When this happens, the agent uses the event to match the ECA (Event-Condition-Action)_rule. According to the matched ECA rules, the event could trigger an action to be executed or a goal to be targeted for achievement. If a goal is to be achieved, the goal, beliefs and Inference rules (I rules) are employed to derive a suitable plan to achieve the goal. When a plan is selected, the actions contained in the plan are scheduled and then executed. The results of the actions that are executed change the VCE and this change may, in turn, result in new events. The concepts or terms used in the reusable agent architecture are defined as follows:

- *Environment*: An agent *environment* is a physical or software place that the agent looks after
- *Goal*: An agent must achieve goals autonomously for its user or other agents. A *goal* is a representation of what the agent user or other agents intend the agent to achieve. An *agent user* (or *user*) is a human who uses the agent
- *Belief*: An agent has a collection of beliefs. A *belief* is a statement that the agent believes to be true. For example, an agent could believe "Peter is a good instructor of subject x". An agent employs beliefs to select plans and actions to execute
- *Event*: An agent can perceives the events happening in the environment that it looks after. An *event* is a signal associated with an occurrence in the environment at a point in time. An event could be "goal_created(Workspace w, Goal g)" that represents a goal created in a workspace. An event could trigger an action to be executed directly and it is called agent *reactive reasoning*
- *Plan:* Plans are used in the intelligent agent to achieve goals. A *plan* is a description of a sequence of actions that an agent can execute when an event occurs such as "goal_decided(Goal g)"
- *Action*: An agent can perform actions that change the environment. An *action* is an operation that can cause a change in the environment
- *ECA Rule*: An ECA (Event-Condition-Action) rule is a statement with the format of *"On an event if the condition is true then do an action or achieve a goal"*
- *I Rule*: An I (Inference) rule is a statement with the format of "if x is true then y is true"

4.4 The Agent Internal Knowledge Base

The agent internal knowledge base is used to represent and store the internal knowledge employed by the agent to achieve its defined goals. The knowledge (as shown in Fig. 8) includes environments, beliefs, ECA rules, I rules, plans and actions as defined in section 4.3. It also includes the agent personal information and multi-agent interaction messages and protocols.

An agent has its personal information that can be assigned by its designer or user. The agent personal information includes:

- *ID*: the unique identifier of an agent in the agent community
- *Name*: the name of the agent
- *Creator*: the name of the person who generates the agent
- *Birthdate*: the date of creation of the agent

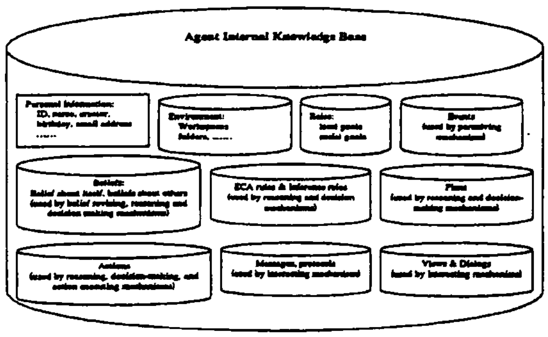

Fig. 8. An agent internal knowledge base

For interacting with other agents, an agent has a message box that represents and stores agent incoming and outgoing messages, and the agent has an interaction protocol repository that represents and stores interaction protocols. Messages could be dropped by the agent or user according to the ECA rules.

4.5 The Implementation of the Reusable Intelligent Agent Architecture

The reusable intelligent agent architecture has been implemented using Java. Fig. 9 is the interface that shows how the ECA rules in an agent are modified dynamically.

Fig. 9. The interface of the intelligent agent. It shows how an ECA rule in the agent is dynamically modified

When an agent joins the OMAS, it starts to detect events. Based on the detected events and the ECA rules, an agent provides autonomous actions that are derived either directly from an ECA rule or from an ECA rules to goal then to plan and then to actions. The agent internal knowledge base can be dynamically change by an agent user when this change is necessary.

5 A Knowledge-Driven Process Management Application

An application for research project management in a university environment is described here to illustrate the applicability of this agent-based collaborative architecture. A research project is typically a knowledge-driven process because its goal and activities and means to achieve that goal may not be precisely specified in advance but emerge over time as knowledge is gained from the actions performed. A specific scenario is described as follows:

Igor (igorh) and Brian (brian) are supervising a collaborative research project in which Alan (alin) and Kevin (kevin) are the research members. The initial goal of the research is to construct an agent-based active knowledge portal for process management. The research project is represented and managed in this application. The project members work together supported by the collaborative architecture. They create their personal workspaces to manage their own knowledge; they publish their ideas, papers, and documents to a collaborative workspace; and they discuss the topics related to the research project within the collaborative workspace. Each member is equipped with a personal intelligent agent that works on behalf of one and only one team member. Those agents can detect events produced in the collaborative workspace and then derive autonomous actions to respond the changes of the collaborative workspace. For example, when the member "igorh" publishes a new document in the workspace, the agent alin detects that event and then produces a piece of knowledge in alin's personal workspace so that member alin can quickly respond to the document published by igorh.

Fig. 10 illustrates the structure of this management scenario. In this structure, each project member has a personal workspace and a personal intelligent agent that looks after a personal workspace and the collaborative workspace. The goal of the agent is to transfer useful knowledge between a personal workspace and the collaborative workspace autonomously based on events detected and actions executed. The knowledge is this application is a document, a paper, an activity, a discussion, or an ECA rule.

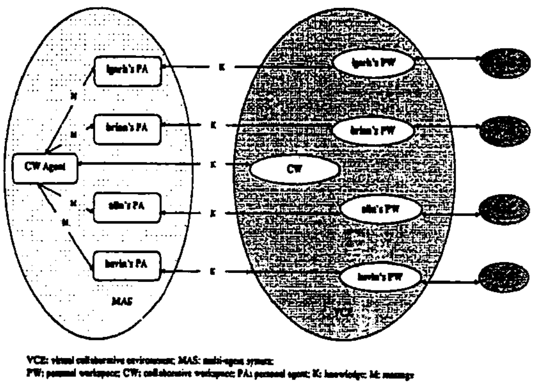

Fig. 10. The application of a research project management built on the agent-based collaborative architecture

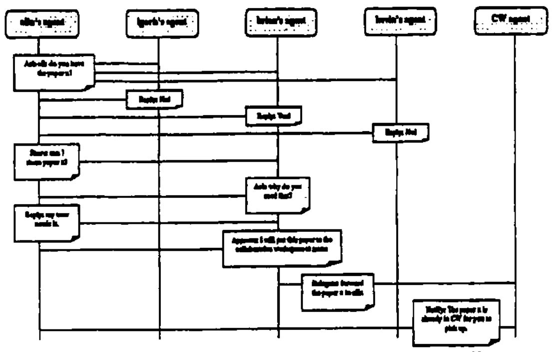

Fig. 11. The message exchanging sequence diagram that describes the message flow between agents in a "sharing" interaction

Fig. 11 shows an interaction between alin's agent and brian's agent because alin's agent wants to share a piece of knowledge that brian's agent owns. In this interaction, primitives "ask-all", "ask", "reply" "share", and "approve" are used. Other primitives such as "refuse" and "not-understand" could be used in a "sharing" interaction protocol.

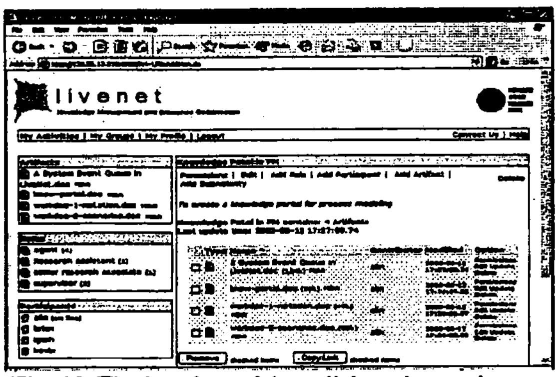

Fig. 12. The interface of the collaborative workspace in this application

Fig. 12 is an illustration of the interface of the collaborative workspace for the activity "Knowledge Portal for Process Modelling". In the left panels, the interface lists all artifacts, roles and participants of workspace. The "agent" is a specific role that has four participants — igorh's agent, brian's agent, alin's agent, and kevin's agent. Some artifacts could be transferred by agents from the participants' personal workspaces to the collaborative workspace.

6 Summary and Future Work

This paper described an agent-based collaborative architecture to support collaborative work. It is applied here to knowledge-driven process management. The collaborative architecture is an integration of an open multi-agent architecture with a virtual collaborative environment. This paper describes the architecture in a high level model, the virtual collaborative component, the open multi-agent component and an application.

A further goal in this research is to provide a methodology and associated tools using Agent UML [2] as a notation for designing reusable agent components so that agents can be constructed efficiently based on the agent architecture.

Acknowledgements

We wish to thank the Australian Research Council for providing funding for building reusable agents in collaborative environment. This is contribution number 03/11 of the Centre for Object Technology Applications and Research.

References

1. Biuk-Aghai, R.P. and Hawryszkiewycz, I.T.: "Analysis of Virtual Workspaces" in Yahiko Kambayashi and Hiroki Takakura (eds.), Database Applications in Non-Traditional Environments '99, Kyoto, Japan, 28-30 November 1999, pp. 325-332, IEEE Computer Society.
2. Bauer, B., Muller, J. P. and Odell, J.: "Agent UML: A Formalism for Specifying Multiagent interaction" in Agent-Oriented Software Engineering, Ciancarini, P., and Wooldridge, M. (eds), Springer, Berlin, pp. 91-103.
3. Debenham J.K.: "Supporting Strategic Process" in proceedings Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents PAAM2000, Manchester UK, April 2000, pp237-256.
4. Finin, T. and Labrou, Y.: "A Proposal for a new KQML Specification" in University of Maryland Baltimore Count (UMBC). Baltimore, 1997.
5. FIPA (Foundation for Intelligent Physical Agents): "Agent Communication Language" in http://www.fipa.org/specs/fipa00003/OC00003A.html, 1998
6. Hawryszkiewycz, I.T.: "Supporting Teams in Virtual Communities" in Bench-Capon, T. Soda, G., Min Tjoa, A. eds, Procededings of the DEXA99 Conference, Florence, September, Springer, Berlin, 1999, pp. 550-558. ISBN 3-540-66448-3.
7. J2EE 1.4 Documentation. http://java.sun.com/j2ee/1.4/docs/
8. Lin, A.: "Multi-agent Business Process Management" in proceedings of ISA'2000, International ICSC Congress on Intelligent Systems and Applications on December 11-15, 2000, University of Wollongong, NSW, Australia
9. Wooldridge, M.: "Intelligent Agents: the Key Concepts, Multi-Agent Applications and Systems II", Springer Verlag, 2001