A Multi-Agent System for Knowledge Delivery in a Software Engineering Environment.

Conference Paper · January 2005

Source: DBLP

3 authors, including:

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS

SEE PROFILE

A Multi-Agent System for Knowledge Delivery in a Software Engineering Environment

Ricardo de Almeida Falbo, Juliana Pezzin, Mellyssa De Martins Schwambach
Computer Science Department, Federal University of Espírito Santo
Fernando Ferrari Avenue, CEP 29060-900, Vitória - ES – Brazil
falbo@inf.ufes.br, juliana_pezzin@hotmail.com, mellyssa_s@hotmail.com

Abstract. Knowledge Management (KM) main goals are to promote growth, communication, preservation and sharing of knowledge. In KM, software agents can be used to connect organizations' members to the knowledge available. Agents can help especially on knowledge filtering and proactive dissemination (knowledge delivery). When KM services are integrated into a Process centered Software Engineering Environment (PSEE), agents can act based on the defined process. They can search and proactively present knowledge items that might be relevant for the developer's current task. This paper presents a multi-agent system developed for supporting knowledge delivery in ODE, a PSEE.

1. Introduction

Software development is a knowledge intensive effort. In order to produce quality software, software organizations have recognized that it is essential to better use their organizational software engineering knowledge. In this context, knowledge has to be systematically collected, stored in a corporate memory, and shared across the organization. Knowledge Management (KM) systems facilitate creation, access and reuse of knowledge, and one of their main goals is to provide relevant knowledge to assist users in executing knowledge intensive tasks.

In KM, software agents can be used to connect organizations' members to the knowledge available [1]. Among other, agents can help on knowledge filtering and dissemination in a proactive manner.

In the context of software development, KM can be used to manage the knowledge and experience generated during software processes. Although every software project is unique in some sense, similar experiences can help developers perform their activities. Reusing knowledge can prevent the repetition of past failures and guide the solution of recurrent problems [2].

But, KM must be embedded in processes. Thus, in the case of software development, KM activities should be integrate into the software process [3]. Since Process-centered Software Engineering Environments (PSEEs) integrate tool support for software development with support for software process modeling and enactment, it is natural to integrate KM facilities into a PSEE [2, 4]. If a software process is defined, it is easier to implement proactive dissemination (knowledge delivery). In this case, based on the process, agents can act in a proactive manner, searching and offering relevant knowledge items for the developer's current task.

In this paper we present a multi-agent system (MAS) developed for delivering knowledge in a PSEE called ODE [5]. ODE has a KM infrastructure that offers services for knowledge creation, capture, retrieval, access, delivery, use, and preservation. The MAS aims to monitor developer's (users of a PSEE) actions, and based on the software process activity being performed, it proactively presents potential relevant knowledge items.

In section 2, we discuss briefly the synergy between KM, PSEEs and agents. Section 3 presents ODE and its KM infrastructure. This section also discusses some problems detected in the first initiatives of using agents to implement knowledge delivery in ODE. To deal with some of those problems, an infrastructure for agent development in ODE, called AgeODE, was built. This infrastructure is presented in section 4. Section 5 presents the MAS developed for proactively disseminating knowledge in ODE. Finally, in section 6 we discuss related works, and in section 7, we report our conclusions.

2. KM, SEEs and Agents: A Synergy

Nowadays, many software organizations have recognized that their main assets are their intellectual capital. In those organizations, staff turnover rates are high, and they face the challenge of sustaining the level of competence needed to compete in the software development market. Knowledge in software engineering is diverse and it grows rapidly. It involves knowledge about technologies, application domains, local policies and practices, among others. In this context, organizations are faced with the problem of providing employees quickly and efficiently with the knowledge required to successfully perform their

tasks. Also, we have to consider that most of the time, team members are making decisions based on their personal knowledge and experience, or knowledge gained using informal contacts. This process is inefficient for large organizations. In fact, software organizations have problems in identifying the content and location of the knowledge, and using it. Thus, an improved use of this knowledge is the main motivation for using Knowledge Management (KM) in software engineering [6, 4].

Although KM has being applied in software engineering for more than ten years, only few implementations are found in current software organizations, due mainly to the lack of a systematic integration into the every-day developer's activities [4]. In fact, to be effective, KM should be integrated into the software process [3]. Since Process-centered Software Engineering Environments (PSEEs) are software systems that assist in the modeling and automation through enactment of software processes [7], they seem to be the most promising platform for integrating KM into the software process. On the other hand, as the complexity of software processes increases, the use of knowledge during software development becomes essential to support software development activities. This claim represents the basis for integrating KM into PSEEs. This way, PSEEs and KM complement each other in order to assist software developers during the software process.

Even when KM is integrated into a PSEE, we have to consider that we still have a problem: knowledge dissemination, especially as the volume of knowledge items grows. In general, we can distinguish between two approaches: knowledge access (passive KM systems) and knowledge delivery (active KM systems) [4, 8]. In a passive KM system, users have to explicitly query it for relevant knowledge items, whenever they have a need. This approach seems to be insufficient for software organizations, because users might be unaware that a relevant knowledge item exists, or they are often too busy to look for it, or they might be unable to query an information system appropriately, among others [4, 9]. In contrast, an active KM system distributes knowledge items to users whenever it is necessary for their work [4]. In fact, knowledge delivery complements the knowledge access approach. While knowledge access is a user-initiated search, knowledge delivery is a system-initiated presentation of knowledge items intended to be relevant to the user's task [8].

In the context of knowledge delivery, agents play an important role. As long as knowledge delivery concerns proactively presenting relevant knowledge that helps workers do their jobs [9], autonomous agents seem to be a very useful approach to deal with this problem. But, to do that, the KM system must be aware about the enactment of the software process. Then, the agents of the KM system should be immersed in a PSEE.

In the next three sections, we discuss how we explore the synergy between PSEE, KM and agents in ODE, a PSEE.

3. ODE: An Ontology-based SEE

ODE (**O**ntology-based software **D**evelopment **E**nvironment) [10] is a PSEE, which is being developed at the Software Engineering Laboratory of the Federal University of Espírito Santo (LabES). It is implemented using only free software, including Java, PostgreSQL and Linux.

As its name indicates, ODE is developed based on some software engineering ontologies, and has several tools, such as tools supporting software process definition, resource allocation, estimation, risk analysis and object modeling, among others.

To support KM in ODE, a KM infrastructure [5] was developed. As shown in Figure 1, the *organizational memory* (OM) is at the core of this infrastructure. Arranged around it, KM services are provided to support the main activities of a general KM process: creation and capture, retrieval and access, delivery, use, and maintenance of organizational knowledge.

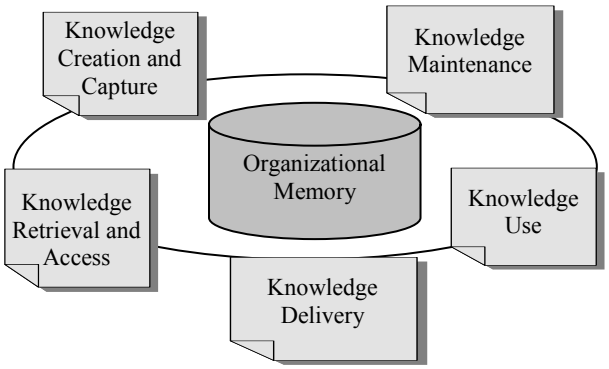

Figure 1 - ODE's KM Infrastructure.

ODE's OM is composed of several knowledge repositories, which store different types of knowledge items that are relevant to software development, including artifacts, lessons learned, and message packages [5].

The KM services are grouped in two categories: general services, which are actually incorporated to ODE as a whole, and tool specific services, which cannot be made available to the environment, because they need to be customized to a specific tool [10]. General services include [10]: (i) *knowledge creation and capture* - offers facilities to capture knowledge items (artifacts, discussion packages and lessons learned); (ii) *knowledge retrieval*

and access - supports access to knowledge items through searching; (iii) *knowledge use* - deals with the feedback about knowledge items' utility; and (iv) *knowledge maintenance* - concerns managing the knowledge repositories based on users' feedback.

Knowledge delivery is the tool specific service, because it is not possible to provide proactive knowledge dissemination without knowing details about the task being done. Thus, it is a service that must be implemented in each tool with KM support [10]. In ODE, agents are being used to implement this service.

The initial proposal was to have agents monitoring the users' actions when they were using specific tools. In this case, agents see what users are doing, and inform them about potential relevant knowledge items. In each tool with KM support, there might be an agent [10] responsible for knowledge delivery. This approach was implemented in some of ODE's tools, namely: quality control [10], resource allocation, and risk management [11]. But, when developing those agents, some problems were detected, such as:

P1. Each agent was built in a different way by each one of the developers. There was neither standardization nor uniformity in agent building, causing integration problems;
P2. Each developer starts from the scratch in the arduous task of building agents. Therefore, there wasn't any form of reuse;
P3. Since the KM system has to track the software process activities, there is also the need for a general agent monitoring the user. This agent should interact with the other agents that act in the specific tools;
P4. Some tools cover complex tasks, and we need a multi-agent system (MAS) acting in this tool, instead of a single agent.

Those problems can be summarized in one: agent integration. Agent integration has to take care about uniform ways of agents communicating, presenting and acting. To deal with agent integration, we built AgeODE, an infrastructure to support the development of agents embedded in ODE.

4. AgeODE: ODE's Infrastructure for Building Agents

Although there are several infrastructures supporting agent building, none of them is bound for building agents embedded in a SEE. Thus, to fulfill this gap in ODE, we developed AgeODE.

AgeODE was defined as a layer over JAT*Lite* [12], using some of its classes, mainly to treat agent communication. Moreover, AgeODE: (i) defines some classes of agents that are potentially useful in the context of SEEs, (ii) defines how communication between agents occurs, and how agents access the objects in the SEE's repository (that is, the objects that are part of their knowledge bases), and (iii) establishes how the agents' internal architecture is.

Agent communication in ODE follows the same client-server model defined in JAT*Lite*: client agents use the routing service offered by a server agent, called router. Thus, specializing the main agent classes of JAT*Lite* (*RouterClientAction* and *RouterAction*), there are two classes of AgeODE: *ClientAg* (Client Agent) and *RouterAg* (Router Agent), as shown in Figure 2.

ClientAg gives to AgeODE's client agents the same features of JAT*Lite*'s client agents, offering services to send and receive other agents' messages. *RouterAg* works as a message router. This type of agent supplies services to name, address and locate agents in a multi-agent system. With a router, agents do not have to know other agents' addresses nor how to communicate with them. These tasks are under the responsibility of the router that works as a communication bridge among the agents linked to it.

The communication protocol used for agent communication in AgeODE is KQML (Knowledge Query and Manipulation Language) [13], since JAT*Lite* already adopts this language. KQML messages in AgeODE, as in JAT*Lite*, are implemented in the following way: each message is a structure that has several fields of the string type, one for each parameter of the message. To compose a KQML message, an agent should fill out the corresponding fields and send it. When receiving a message from another agent, the receiver agent interprets it according to the guidelines of KQML.

Being a generic infrastructure for agent development, JAT*Lite* does not define other classes of agents; it only separates them into clients and server. However, in the context of SEEs, it is interesting to provide a basic set of agent classes including features that are useful in several situations in a SEE. Thus, four classes of client agents were proposed for AgeODE, as shown in Figure 2.

Interface Agents (*InterfaceAg*) aim to offer to SEE's users a friendlier interface, with proactive characteristics. An Interface Agent detects the users' actions when using an interface of the environment (or of one of its tools), and based on that, they act.

An User Agent (*UserAg*) uses the knowledge that it has about a certain user to support him on performing his tasks. It should be able to establish the user's profile, looking for relevant features of the user. An user agent typically interacts with the user that it represents, and aids him to do tasks and to make decisions.

Information Agents (*InformationAg*) are responsible for performing some system functionality. Its main task is to look for information and to accomplish tasks inside the SEE.

Finally, the Coordinator Agent (*CoordinatorAg*) aims to coordinate the tasks being executed at a given moment by a set of agents in the SEE. For such, it should be able to distribute tasks to agents, to consolidate results of tasks, to retrieve information from one or more dispersed agents in the society, and to know the agents (and their specific capabilities/abilities) that are under its coordination domain.

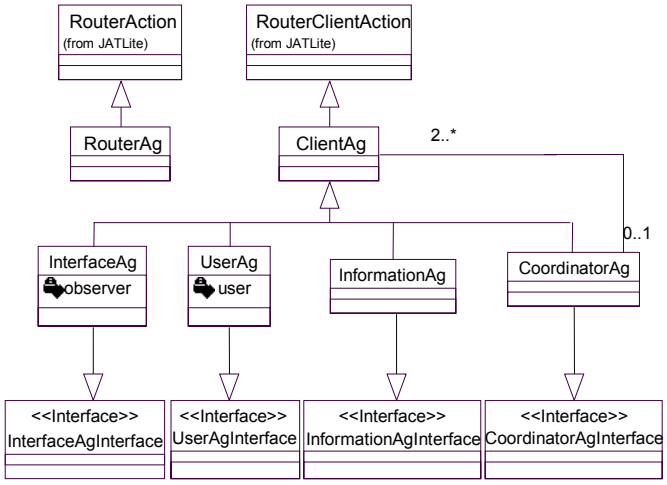

Figure 2 - Agent Classes in AgeODE.

Since a concrete client agent for an ODE's application can be of several types, there are also interfaces associated to the client agent types of AgeODE. This way, if an agent has features of both an Interface Agent and an User Agent, then it can be implemented, for example, inheriting from the *InterfaceAg* class and realizing the *UserAgInterface* interface.

Finally, it is worthwhile to point out that, because an Interface Agent has to capture events from the SEE's user interfaces (UI), we need to establish a way to agents monitor these UIs. In AgeODE, it is done by observer objects that are associated to interface agents. Since agents and the SEE are isolated computational processes, we decided to designate to observer objects the responsibility for monitoring UI events. Observers run in the SEE and intercept UI events, sending via sockets, messages to its corresponding Interface Agent that becomes aware about the user's actions and acts properly. This way, observers act as the Interface Agent's perception mechanism in the environment, capturing UI events in a way that the agents and the SEE are actually implemented as separated computational processes.

Using AgeODE, two aspects of the agent integration problem (P1 and P2) listed in section 3 were treated. But, we also need to deal with the other two aspects (P3 and P4). To do that, we established a multi-agent system (MAS) basic architecture for knowledge delivery in ODE, which is discussed next.

5. A MAS for Knowledge Delivery in ODE

As previously mentioned, in ODE, knowledge delivery is implemented using agents. Each tool with knowledge delivery facilities must have an agent or a MAS acting in it (P4). Also, there must be some general agents that are useful for all the tools (P3). To deal with these requirements, we proposed a MAS general architecture for knowledge delivery that consists of three general agents, besides the tool specific agents: the Personal Assistant Agent, the ODE's Router Agent, and the Similar Project Identifier Agent.

The Personal Assistant Agent (*PersonalAssistantAg*) accompanies an ODE's user since the moment he accesses the environment until the moment he leaves it. This agent knows the software process, and the tools that can be used in each one of its activities. Moreover, it knows the user, and establishes his profile in the environment, allowing the user to access the tools that he was using the last time he used ODE. This agent also knows the specific agents of each tool, if they exist, and it is responsible for starting these agents when a tool is initiated. *PersonalAssistantAg* is implemented inheriting from AgeODE's *InterfaceAg* class and realizing *UserAgInterface* and *CoordinatorAgInterface*.

The ODE's Router Agent (*ODERouterAg*) is responsible for agent communication in ODE. As discussed before, communication between agents in AgeODE requires a Router Agent. This is the role of *ODERouterAg*, which inherits from *RouterAg*.

Finally, since in KM, in general, it is very important to identify similar past projects to present relevant knowledge items, there is an agent, called Similar Project Identifier Agent (*SimilarProjectIdAg*), which is responsible for identifying similar projects to a given one. It is a subtype of AgeODE's *InformationAg*, and all agents that need to know about similar projects must interact with it.

Beyond these three agents, each tool with knowledge delivery services has to have its own agent or MAS, according to the complexity of the task being supported. When a MAS is necessary, one of its agents must be a Coordinator Agent. This coordinator is responsible for coordinating the tool's internal agents, and also for interacting with the *PersonalAssistantAg* and with the *SimilarProjectIdAg*.

This approach was followed to reengineer the knowledge delivery services of two ODE's tools: human resource allocation and risk management, as shown in Figure 3. The first one has only one agent acting in it, since the task is relatively simple. The second has a MAS embedded in it, because it supports a more complex activity that, in fact, is decomposed into sub-activities with some complexity.

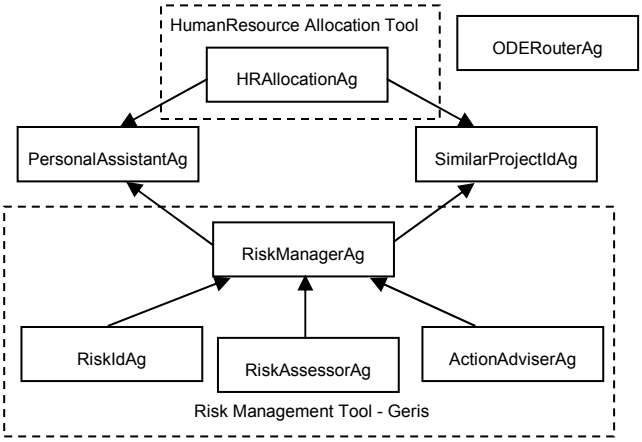

Figure 3 - ODE's MAS for Proactive Knowledge Dissemination.

In the Human Resource Allocation Tool, the Human Resource Allocation Agent (*HRAllocationAg*) supports the task of allocating human resource to project activities. This agent suggests the resources to be allocated for a specific activity, based on the project team, the competencies of each member and past allocations already done in similar past projects. Because it is responsible for aiding to perform a task, it is an *InformationAg*. But it has also to monitor the tool's interface. Then, it realizes the *UserAgInterface*. The complexity involved in this task is not so high and, then, only an agent acts in this tool. This agent interacts with the *PersonalAssistantAg*, and since it uses similar past projects to give its suggestions, it also interacts with the *SimilarProjectIdAg*.

To support knowledge delivery in risk management, there is a MAS composed of four agents, as shown in Figure 3. This MAS is embedded in GeRis, the ODE's risk management tool [11]. GeRis supports a risk management process composed of the following activities [11]: (i) risk identification - attempts to establish risks to the project; (ii) risk analysis - concerns analyzing the identified risks, estimating probability of occurrence and impact; (iii) risk assessment - aims to rank the identified risks and to establish priorities; (iv) action planning - concerns planning mitigation and contingency actions for the managed risks; and (v) risk monitoring - consists of redoing the activities above as the project proceeds.

In GeRis' MAS, agents were designed to support specific activities of the risk management process. The Risk Identifier Agent (*RiskIdAg*) acts during risk identification. It suggests which risks should be identified for the project, based on similar past projects. The Risk Assessor Agent (*RiskAssessorAg*) acts during risk analysis and evaluation. It supports the assessment of risks impact and probability, and also supports the definition of which risks should be managed in the project. In both cases, this agent uses information of similar past projects. At last, the Action Adviser Agent (*ActionAdviserAg*) acts in action planning. It suggests contingency and mitigation actions to be taken to treat risks, also based on similar past projects. All these agents (*RiskIdAg*, *RiskAssessorAg* and *ActionAdviserAg*) are Information Agents.

Since we have a MAS acting in risk management, we need a coordinator agent to coordinate their actions. This is the role of the Risk Manager Agent (*RiskManagerAg*), which is a *CoordinatorAg*. It is considered the main agent of the risk management tool and the only one in this tool that is known by the *PersonalAssistantAg*. When GeRis is initiated, the *PersonalAssistantAg* starts this agent. It is, in turn, responsible for starting the other risk management agents based on the activity of the risk management process that the user is performing. To do that, it has to monitor GeRis' UI, and then, it also realizes the *UserAgInterface*. Moreover, since all the other agents need information about similar past project, the *RiskManagerAg* interacts with the *SimilarProjectIdAg*, capturing the past projects that are similar to the current being performed.

The agent-based knowledge delivery approach applied in these two tools reflects the general approach defined to implement knowledge delivery in ODE. Every tool in which we want to implement knowledge delivery facilities needs to have an agent or a MAS associated to it. If the activity being supported by the tool is complex, a MAS is preferred. In this case, we always need to have a coordinator agent as the tool's main agent, and the Personal Assistant Agent has only to know it.

Finally, we should highlight that this approach is strongly supported by AgeODE. Each agent class is implemented as one of the agent types defined in it, and sometimes realizes another agent type interface.

6. Related Work

There are several works in the literature describing the use of agents for knowledge management (KM) (see [14]), and some approaches for integrating KM and PSEE, some of them exploring knowledge delivery. However, we did not find an approach exploring the

synergy between agents, KM and PSEE. Let's examine some work done in integrating KM and PSEE.

Santos et al. [15] explores the concept of Enterprise-Oriented SEE (EOSEE), matching KM with PSEE. As ODE, EOSEEs are based on ontologies. But Santos et al. say nothing about knowledge delivery.

Holz [4] attacks the problem of delivering knowledge in software organizations by means of an approach that represents recurrent information needs associated with appropriate software process assets, and retrieves the information in a two-phase, interactive retrieval model. This approach was implemented in a system called PRIME, which was coupled with the MILOS PSEE. As in ODE, PRIME provides developers with relevant information. The main difference is that in PRIME, a list of pre-defined information needs is presented, and, from this list, developers can choose one, and trigger an automatic retrieval of information. In ODE, agents try to capture this information needs and notify the user that they have some useful knowledge items or suggestions. As in PRIME, ODE's users are free to inspect or not the items suggested.

7. Conclusions

For the successful enactment of software processes, it is essential that developers are provided just-in-time with knowledge items that are relevant and useful for their current tasks [4]. Thus, knowledge delivery is becoming more and more important. In this paper, we presented an agent-based approach used to integrate knowledge delivery facilities into ODE, a Process-centered SEE. This approach consists of developing specific agents to deal with the information needs of activities of the software process. Also an infrastructure for building agents embedded in the environment, called AgeODE, was developed in order to deal with agent integration in ODE.

Although ODE is being used in a software house, we do not perform a deep evaluation of the appropriateness of the support being provided by the agents yet. The initial results regarding the use of the tools with knowledge delivery support are promising. But we expect that, based on the users' feedback, we can refine the agents' behavior in order to better support those activities.

Acknowledgments

This work was accomplished with the support of CNPq, an entity of the Brazilian Government reverted to scientific and technological development.

References

[1] O'Leary, D.E., "Enterprise Knowledge Management", IEEE Computer, 54-61, March 1998.

[2] A.C.C. Natali, R.A. Falbo, "Knowledge Management in Software Engineering Environments", In: Proceedings of the XVI Brazilian Symposium on Software Engineering - SBES'2002, 238-253, Gramado, Brazil, October 2002.

[3] S. Henninger, "Using Software Process to Support Learning Software Organizations", in Proceedings of the Workshop on Learning Software Organizations – LSO'1999, 99-114, Kaiserslautern, Germany, June 1999.

[4] H. Holz, Process-Based Knowledge Management Support for Software Engineering, Doctoral Dissertation, University of Kaiserslautern, dissertation.de Online-Press, 2003.

[5] R.A. Falbo, D.O. Arantes, A.C.C. Natali, "Integrating Knowledge Management and Groupware in a Software Development Environment", in Proc. of the 5th International Conference on Practical Aspects of Knowledge Management, 94-105, Vienna, Austria, 2004.

[6] I. Rus, M. Lindvall, "Knowledge Management in Software Engineering", IEEE Software, 26-38, May/June 2002.

[7] S. Arbaoui, J.C. Derniame, F. Oquendo, H. Verjus, "A Comparative Review of Process-Centered Software Engineering Environments", Annals of Software Engineering 14, 311-340, 2002.

[8] G. Fischer, J Ostwald, "Knowledge Management: Problems, Promises, Realities, and Challenges", IEEE Intelligent Systems, 60-72, January/February 2001.

[9] A. Abecker, et alli, "Towards a Technology for Organizational Memories", IEEE Intelligent Systems, 40-48, May/June 1998.

[10] R.A. Falbo, A.C.C. Natali, P.G. Mian, G. Bertollo, F.B. Ruy, "ODE: Ontology-based software Development Environment", in Proc. IX Argentine Congress on Computer Science, 1124-1135, La Plata, Argentina, 2003.

[11] R.A. Falbo, F.B. Ruy, G. Bertollo, D.F. Togneri, "Learning How to Manage Risks Using Organizational Knowledge", Advances in Learning Software Organizations, Melnik G. and Holz, H. (Eds.): LNCS 3096, 7-18, 2004.

[12] H. Jeon, C. Petrie, M.R. Cutkosky, "JATLite: A Java Agent Infrastructure with Message Routing". IEEE Internet Computing, March /April 2000.

[13] T. Finin, Y. Labrou, J. Mayfield, "KQML as an agent communication language", Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM94), ACM Press, December 1994.

[14] V. Dignum, "An Overview of Agents in Knowledge Management", Institute of Information and Computing Sciences, Utrecht University, technical report UU-CS-2004-017, 2004.

[15] G. Santos, K. Villela, L. Schnaider, A.R. Rocha, G.H. Travassos, "Building Ontology-based Tools for a Software Development Environment", Advances in Learning Software Organizations, Melnik G. and Holz, H. (Eds.): LNCS 3096, 19-30, 2004.