A Multi-Agent System to Support Remote Software Development

Marco Mari, Lorenzo Lazzari, Alessandro Negri, Agostino Poggi and Paola Turci

***Abstract*—In this paper, we present a Web and multi-agent based system to support remote students and programmers during common projects or activities based on the use of the Java programming language. This system, called RAP (Remote Assistant for Programmers), associates a personal agent with each user. A personal agent helps its user to solve problems proposing information and answers, extracted from some information repositories, and forwarding answers received from other "on-line" users, that were contacted because their personal agents recommend them as experts in that topic. To be able to recommend their users, personal agents build and maintain a profile of them. This profile is centered on user's competences and experience and is built by extracting information through both the code she/he wrote and the positive answers the user gave to other users. A first prototype of the system is under implementation in Java by using the JADE multi-agent development framework. This prototype will be tested in practical courses on JADE shared among students of some American Latin and European Universities inside the European Commission funded project "Advanced Technology Demonstration Network for Education and Cultural Applications in Europe and Latin America".**

***Index Terms*—Cooperative systems, multi-agent systems, intelligent tutoring systems.**

I. Introduction

FINDING relevant information is a longstanding problem in computing. Conventional approaches such as databases, information retrieval systems, and Web search engines partially address this problem. Often, however, the most valuable information is not widely available and may not even be indexed or cataloged. Much of this information may only be accessed by asking the right people. The challenge of finding relevant information then reduces to finding the "expert" whom we may ask a specific question and who will answer that question for us. However, people may easily get tired of receiving banal questions or different times the same question, therefore, who needs help for solving a certain problem, should look for documents related to the problem and then eventually look for a possible expert on the topic.

This kinds of problems are very relevant in the software development because of the wide variety of software solutions, design patterns and libraries makes hard to take the best decision in every software development phase, and a developer can't always have the required expertise in all fields.

In this paper, we present a multi-agent based system, called RAP (Remote Assistant for Programmers), that integrated information and expert searching facilities for communities of student and researchers working on related projects or work and using the Java programming language. In the following section, we describe the RAP system, the current state of its implementation and some preliminary evaluation results, then we introduce related work and, finally, we give some concluding remarks and present some our future research directions to improve the RAP system.

Manuscript received September 27, 2004. This work is partially supported by the European Commission through the contract "@lis Technology Net (ALA/2002/049-055)" and by "Ministero dell'Istruzione, dell'Università e della Ricerca" through the COFIN project ANEMONE.

M. Mari is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905712; e-mail: mari@ce.unipr.it).

L. Lazzari is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905712; e-mail: lazzari@ce.unipr.it).

A. Negri is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905712; e-mail: negri@ce.unipr.it).

A. Poggi is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905728; e-mail: poggi@ce.unipr.it).

P. Turci is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905708; e-mail: turci@ce.unipr.it).

M. Baldoni, F. De Paoli, A. Martelli, and A. Omicini (Eds.): *WOA 2004 – Dagli Oggetti agli Agenti, Sistemi Complessi e Agenti Razionali*, Torino, Italy, November 29th – December 1st 2004. Pitagora Editrice Bologna. ISBN 88-371-1533-4. http://woa04.unito.it

II. The RAP System

RAP (Remote Assistant for Programmers) is a system to support communities of students and programmers during shared and personal projects based on the use of the Java programming language. RAP associates a personal agent with each user which helps her/him to solve problems: proposing information and answers extracted from some information repositories, and forwarding answers received by "experts" on the topic selected on the basis of their profile. A personal agent also maintains a user profile centered on her/his competences and experience built through the positive answers given to other users and by extracting information through the code she/he has written.

A. System Agents

The system is based on seven different kinds of agents: Personal Agents, Code Documentation Managers, Answer Managers, User Profile Managers, Email Manager, Starter Agent and Directory Facilitators.

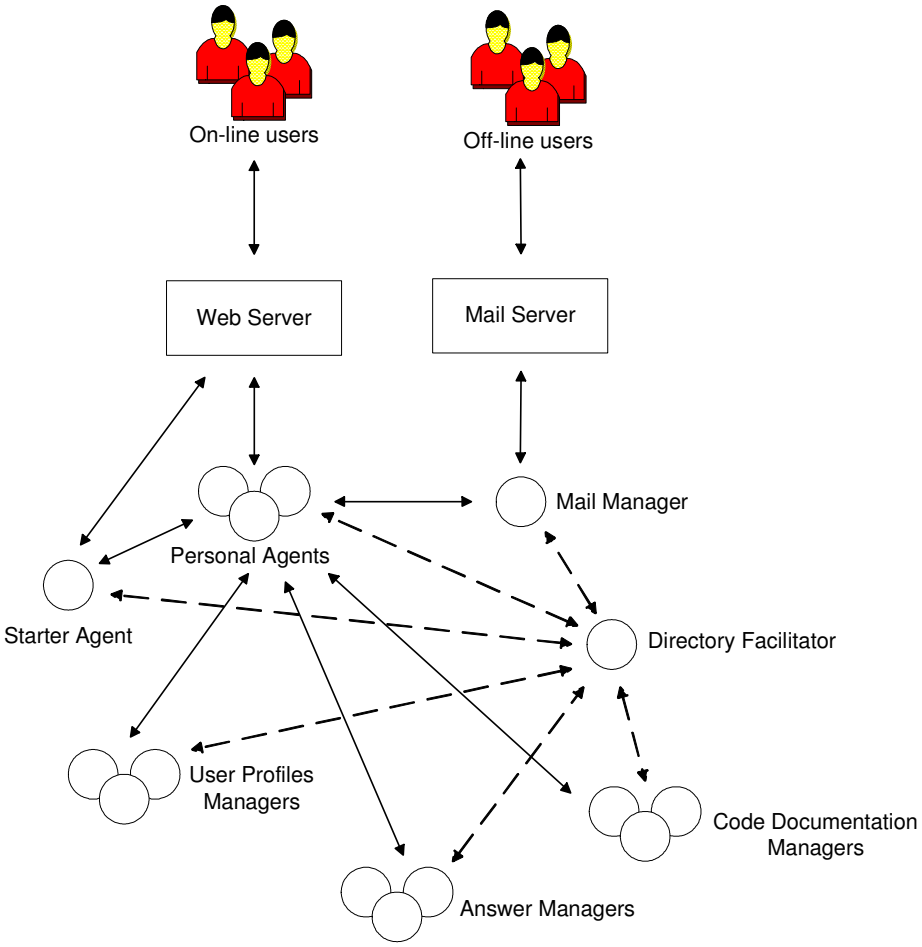

Fig. 1. RAP platform architecture.

Personal Agents are the agents that allow the interaction between the user and the different parts of the system and, in particular, between the users themselves. Moreover, this agent is responsible of building the user profile and maintaining it when its user is "on-line". User-agent interaction can be performed in two different ways: when the user is active in the system, through a Web based interface; when it is "off-line" through emails. Usually, there is a personal agent for each on-line user, but sometimes personal agents are created to interact with "off-line" users via emails.

User Profile Managers are responsible of maintaining the profile of "off-line" users and of activating personal agents when it is necessary that they interact with their "off-line" users via emails.

Code Documentation Managers are responsible of maintaining code documentation and of finding the appropriate "pieces of information" to answer the queries done by the users of the system.

Answer Managers are responsible of maintaining the answers done by users during the life of the system and of finding the appropriate answers to the new queries of the users. Besides providing an answer to a user, this agent is responsible of updating the score of the answer and forwarding the vote to either the personal agent or the user profile manager for updating the profile of the user that performed such an answer.

Email Managers are responsible for receiving emails from "off-line" users and forwarding them to the corresponding personal agents.

Starter Agents are responsible for activating a personal agent when either a user logs on or another agent request it.

Directory Facilitators are responsible to inform an agent about the address of the other agents active in the system (e.g., a personal agent can ask about the address of all the other personal agents, of the code documentation managers, etc.).

Figure 1 gives a graphical representation of a RAP platform and the interactions of personal agents and of the directory facilitator with the other agents of the platform. Note that a RAP platform can be distributed on different computation nodes and that a RAP system can be composed of different RAP platforms connected via Internet. Moreover, in figure 1 groups of three users or agents means that there can be one or more users and agents. Finally, in A RAP system there is a directory facilitator for each platform.

B. System Behavior

A quite complete description of the behavior of the system can be given showing the scenario where a user asks information to its personal agent to solve a problem in its code and the personal agent finds one (or more) "pieces of information" that may help her/him. The description of this

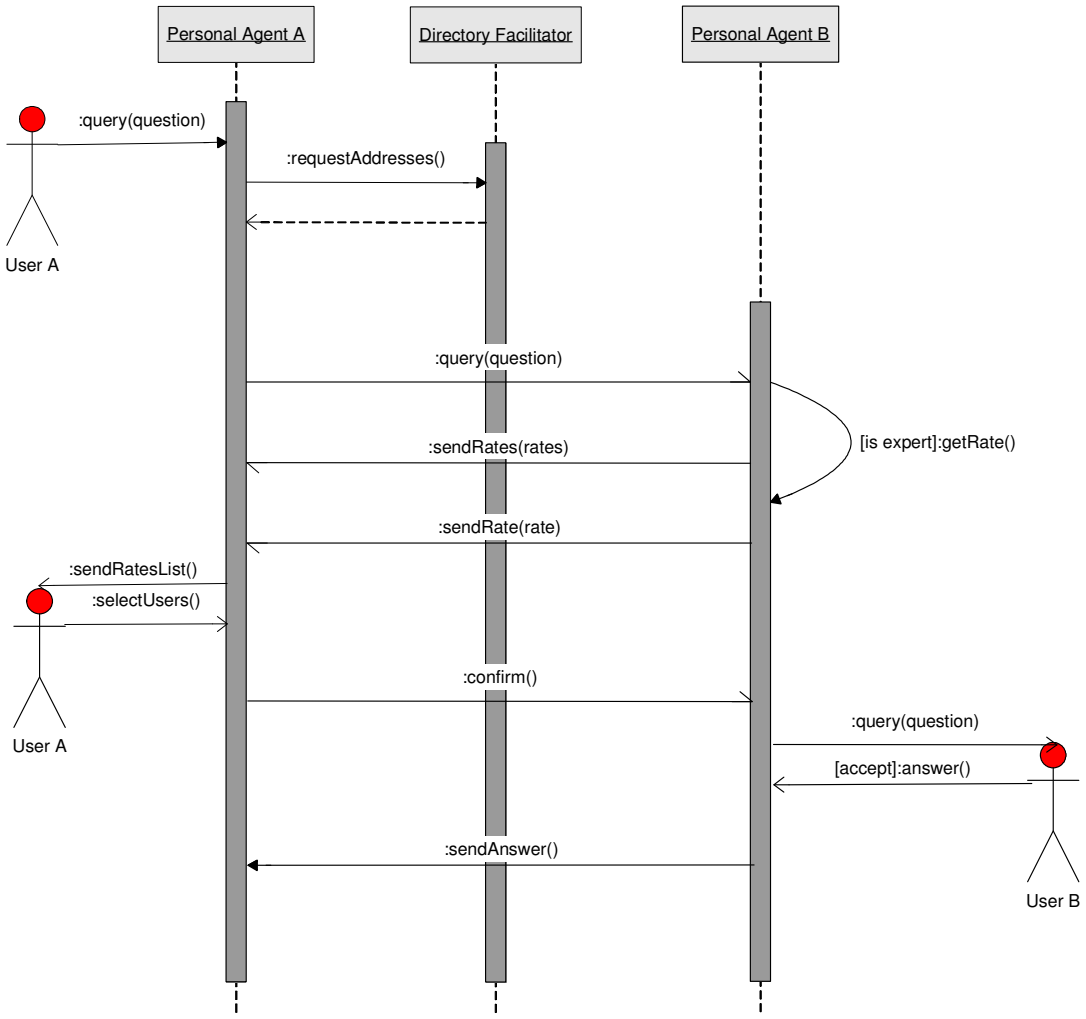

Fig. 2. UML interaction diagram describing how works to allow a user to ask a question and then receive the relative question from an "expert".

scenario can be divided in the following steps:

1) Select answer types
2) Submit a query
3) Find answers
4) Rate answer

Select answer types: the user can receive information extracted from code documentation, answers extracted from the answer repositories and new answers sent by the other users of the system. Therefore, before submitting the query, the user can select the types of answers (one or more) she/he likes to receive.

Submit a query: the user, through its user interface, provides the query to its personal agent. In particular, the user can query either about a class or an aggregation of classes for implementing a particular task or about a problem related to her/his current implementation. The query is composed of two parts. The first part (we call it "annotation") identifies the context of the query and can contains keywords provided by a system glossary and/or the identification of classes and/or methods in a univocal way (i.e., the user needs to specify the complete package name for a class and adds the class name for a method). The second part contains the textual contents of the query.

Find answers: the personal agents perform different actions and interact with different agents to collect the various types of answers.

For getting code documentation, the personal agent asks the directory facilitator about all the code documentation managers. After receiving this information, the personal agent forwards the query to all these agents. These agents search "pieces" of code documentation related to the query and send them to the personal agent associating a score with each "piece".

For getting answers from the answer system repositories, the personal agent asks the directory facilitator about all the answer managers. After receiving this information, the personal agent forwards the query to all these agents. These agents search answers related to the query and send them to the personal agent associating a score with each answer.

The reception of new answers from the system users is a more complex activity and its description can be divided in four further steps (Figure 2 shows the UML interaction diagram describing these phases):

3.1) Find experts
3.2) Receive experts rating
3.3) Select experts

3.4) Receive answers

Find experts: the personal agent asks the directory facilitator about the other active personal agents (i.e., the personal agents of the user that are "on-line") and all the user profile managers of the system (i.e., the agents managing the profile of the users that are not "on-line"). After receiving this information, the personal agent forwards the query to these personal agents together to the user profile managers.

Receive expert rating: all these agents (personal agents and user profile managers) compute the rating of their users to answer to this query on the basis of the query itself and of the user profile. The agents that compute a positive score (i.e., its user may give an appropriate answer to the query) reply to the querying personal agent with the rating of its user (in the case of a personal agent) or its users (in the case of user profile manager).

Select experts: the personal agent divides on-line and off-line users, orders them on the basis of their rating and, finally, presents these two lists to its user. The user can select more than one user and then the personal agent sends the query to the corresponding personal agents (for the "on-line" users) and to the corresponding user profile managers (for the "off-line" users).

Receive answers: the replying personal agents immediately present the query to their user and forward the answer as soon as the user provides it. User profile manager activates the personal agents of the involved users through the starter agent. These personal agents forward the query to their user via email and then terminate themselves. Users can answer either via email or when they log again on the system. In the case of email, the email manager starts the appropriate personal agent that extracts the answer from the email and forwards it. When the querying personal agent receives an answer, it immediately forwards it to its user.

Rate answers: after the reception of all the queries, or when the deadline for sending them expired, or, finally, when the user has already found an answer satisfying its request, the personal agent presents the list of read answers to its user asking her/him to rate them. After the rating, the agent forwards each rating to the corresponding personal agent, code documentation manager, answer manager or user profile manager that provides to update the user profile and/or the answer rating (when a user rates an answer retrieved from the answer repository, this rating is also used to updated the user profile of the user that previously proposed the answer). Note that in the case of rating of users answers, the rating cannot be known by the user that sent the answer and users that did not send answers automatically received a negative rating.

C. User and Document Profile Management

In our system, the management of user and document profiles is performed in two different phases: an initialization phase and an updating phase. Figure 3 gives a graphical description of this process.

In order to simplify, speed up and reduce the possibility of inaccuracy due to people's opinions of themselves and to incomplete information, we decided to build the initial profile of the users and documents in an automated way that, for the users, is very similar to the one used by Expert Finder system [21]. Profiles are represented by vectors of weighted terms whose value are related to the frequency of the term in the document or to the frequency of the use of the term by the user. The set of terms used in the profiles is not extracted from a training set of documents, but corresponds to those terms included in the system glossary, provided to the users for annotating their queries, and to the names of the classes and methods of the Java software libraries used by the community of the users of the system.

While document profiles are computed by using term frequency inverse document frequency (TF-IDF) [19] and profiles weighted terms correspond to the TF-IDF weight, each user profile is built by user's personal agent through the analysis of the Java code she/he has written. In this case, the weight of the terms in the profile corresponds to the frequency is not the TF-IDF weight, but the real frequency of the term in the code of the user (i.e., term frequency is not weighted on the basis of the frequency of the term in the code written by all the users). We used this approach for different reasons. First, we speed up and reduce the complexity of building user profiles. As a matter of fact, TF-IDF algorithm can be easily used in a centralized system where all the profiles and the data to build them are managed. Our context is more complex: the system is distributed, only the personal agent can access to the software of its user, for privacy and security reasons, and the profiles are maintained by the corresponding personal agents or by possibly different user profile managers when the personal agent is not alive. The second and most important reason is that the profile built by personal agents is only the initial user's profile. And it will be updated when the user writes new software and especially when the user helps other users answering their queries.

The updating of user and document profiles is done in three cases: i) a user asks about a problem and then rates some of the received answers, ii) a new software library is introduced in the ones used by the community or some new terms are introduced in the system glossary, and iii) a user writes new software.

In the first case, there are three possible behaviors according to the source of the answer (user, document repository or answer repository).

If the answer comes from a user, on the basis of the received rating her/his profile is updated, of course, only the part concerning the terms involved in the query annotation. Moreover, if the rating is positive, the answer is added to the answer repository and its profile is built from the query annotation and the rating of the answer.

If the answer comes from the document repository and the rating is positive, the answer is added to the answer repository, its profile is the original document profile updated by the rating of the answer.

Finally, if the answer comes from the answer repository and

the rating is positive, the part of the answer profile related to the terms involved in the query annotation is updated on the basis of the received rating. Moreover, in the case that this positive rated answer comes from a user and not from the document repository, also the part of the user profile related to the terms involved in the query annotation is updated on the basis of the received rating. Finally, the query corresponding to such positive rated answer is added in the repository (i.e., the answer was good for one or more previous queries, but also for the current one; queries are ordered by answer rating).

We decided to avoid the use of negative rates for updating the profile of the answers in the answer repository. In fact, if an answer is in the repository, it means that at least a user considered useful to solve her/his problem; therefore, if later on this answer received a negative rate it does only means that the answer is not appropriate for the last query, but it is still appropriate for the previous queries for which it received positive rates.

When a new software library is introduced in the list of libraries used by the users of the system or some new terms are introduced in the system glossary, all the document and user profiles must be updated. While document profiles are rebuilt on the basis of the new complete set of terms, user profiles are updated adding the weighted terms corresponding to the new term, of course with a weight equal to their frequency in the software written by the user.

Finally the user's profile is updated, adding only the new weighted terms, even when the user writes new software.

D. System Implementation and Experimentation

A first prototype of the RAP System is under development by using JADE [3]. JADE (Java Agent Development framework) is a software framework to aid the realization of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [6]. JADE is an Open Source project, and the complete system can be downloaded from JADE Home Page [9].

Given the distributed nature of JADE based agent systems, a RAP system can be distributed on a set of agent platforms connected usually via Internet and situated in different parts of the world. Each agent platform can be distributed on different computation nodes and is connected to a Web server, for allowing direct interactions with the users, and to a mail server, for allowing email interactions with the users. In each agent platform there is a unique starter agent and email agent, but there might be more than one user profile manager, code documentation manager, answer manager. This usually happens when the agent platform is distributed on different nodes in order to cope with performance issues due to the number of the users to be managed. Furthermore, distribution of a RAP platform on different computation nodes and agents replication can be introduced for reliability reasons (in this case, agents manage copies of data) too. Finally, there can be one or more directory facilitators. In the case of more than one directory facilitator, these agents build a hierarchical map of the agents of the system; therefore, when an agent is created, the only information it needs to know is simply the address of the main (root) directory facilitator.

A large part of the first prototype of the system has been completed. In particular, the subsystem supporting interactions among personal agents and the interaction between each pair of personal agent and "on-line" user has been completed. This subsystem has been used with success by a small set of students, connected by different labs or from their house, for the development JADE software within some course works. In these activities, students could formulate queries annotating it with terms extracted from a glossary derived from the Sun "Glossary of Java Related Terms" [20] and class and method names extracted from Java and JADE source code.

Moreover, we have evaluated the system with a simulation. We have asked 10 queries on Java programming to 10 students with experience in Java programming, but with advanced experience on different application fields and software libraries. Of course, the 10 queries were defined in order to put in the light the difference in the knowledge of the students involved. Then, we have evaluated the part of the RAP system involving only user interactions (no document and answer repository). A personal agent is responsible to perform the 10 queries and other 10 personal agents to provide the answers written by the different students. The evaluation has concerned mainly the comparison of the ordered list (built ordering experts on the basis of their profiles) provided by the querying agent to its user with an ordered list of the answers we did before performing the simulation. The simulation was reiterated a number of times equal to the possible orders of the query and the users profiles were reset each simulation. The initial user's profile was built on the basis of the content of the answers associated with this virtual user. Clearly, it cannot be considered a simulation of the real behavior of the system, but the obtained results have encouraged us in the completion of the system. In fact, the differences between the personal agent ordered list and our a priori ordered list decreases during the evaluation process and for the last query we have had an average error of the 5%.

As from the end of this year, the final RAP system will be tested in practical courses on JADE shared among students of some American Latin and European Universities inside the European Commission funded project "Advanced Technology Demonstration Network for Education and Cultural Applications in Europe and Latin America (@lis Technology Net)" [1]. Moreover, the system will be used by students and researchers, involved in the ANEMONE project [2], for cooperating in the realization of agent-based software.

III. RELATED WORK

In the last years a lot of work has been done in the fields of document and expert recommendation and in the development of tools and systems for supporting e-learning and, in particular, computer programming activities.

With the advent of the Web, document recommendation systems are become one of most important area of both

research and application of information technologies. All the most important proposed systems are applied to the recommendation of Web pages and are not specialized for computer programming documents, but usually allow the customization for different subjects. GroupLens is the first system that used collaborative filtering for document recommendation [18]. This system determinates similarities among users and then is able to recommend a document to a user on the basis of the rating of similar users on the recommend document. Syskill &Webert is a system with the goal of helping users distinguishing interesting Web pages on a particular topic from uninteresting ones [16]. In particular, this system recommends document to a user on the basis of her/his user profile that it builds and updates by using user's evaluations of the interestingness of the read documents. Adaptive Web Site Agent is an agent-based system for document recommendation [17]. This system works on the documents of a Web site and recommends documents to visitors integrating different criteria: user preferences for the subject area, similarity between documents, frequency of citation, frequency of access, and patterns of access by visitors to the web site.

Several prior systems support expertise recommendations. Vivacqua and Lieberman [21] developed a system, called Expert Finder, that recommends individuals who are likely to have expertise in Java programming. This system analyzes Java code and creates user profiles based on a model of significant features in the Java programming language and class libraries written by the user. User profiles are then used to assist novice users in finding experts by matching her/his queries with user profiles. A group of researchers at MITRE has also developed an expertise recommendation system called Expert Finder [11],[12]. This system finds experts by performing a query over a MITRE wide corporate database that includes information about 4500 MITRE employees. The entries in the database are manually maintained by each individual employee. After performing the query, the system filters the results and presents a list of employees who are likely to have some expertise in the queried topic. Expertise Recommender is another system that recommend people who are likely to have expertise in a specific area [13],[14]. A user garners recommendation from ER by picking a relevant identification heuristic, selecting a matching technique, and entering a description or terms related to a problem. Then, the system responds with a list of individuals who are likely to have expertise with the problem and who are a good social match for the person making the request. In this system, user profiles are built by processing user‘s day-to-day work products. MARS is a referral system based on the idea of social network [13]. This system is fully distributed and includes agents who preserve the privacy and autonomy of their users. These agents build a social network learning models of each other in terms of expertise (ability to produce correct domain answers), and sociability (ability to produce accurate referrals), and take advantage of the information derived from such a social network for helping their users to find other users on the basis of their interests.

A lot of work has been also done in the development of tools and systems for supporting e-learning and, in particular, computer programming activities. Hazeyama and Osada realized a system for collaborative software engineering education [7]. This system provides both generic collaboration services, useful in all the different phases of students course project, and services dedicated to a specific phase of such a project. In fact, the system offers a bulletin board subsystem and a notification service used by students and teachers along all the project, and, for example, provides a subsystem supporting students code inspection process: this subsystem provides a tool that allows to a teacher the annotation of students code with comments, and manages the interaction between the teacher and the students in the different phases of the inspection process (i.e., code submission, teacher feedback, updated code submission, etc.). WBT (Web Based Teaching) is an agent based collaborative learning support system providing community Web services [8]. The system is centered on a Web site containing teaching materials for computer programming practice and an electronic bulletin board system for question answering to assist students during their programming practice activities. In this system agents have the duty of distributing questions to the teacher or to "on-line" students that previously answered to similar questions. Mungunsukh and Cheng proposed an agent based learning support system for novice programmers in a distance-learning environment [15]. This system is dedicated to the learning of the VLB programming language and its activity can be divided in two phases: student observation and student support. In the first phase, the system attempts to understand students' behavior by observing their typing events, behaviors on different purpose of web browser of lessons, tasks and examples, error types made by students and debugging events on a programming editor. After the acquisition of information about the activities of the students, the system supports students with relevant information as, for instance, related examples and lessons for the problems they are working on, and problems which have similar solutions. I-MINDS is a multi-agent system that enables students to actively participate in a virtual classroom rather than passively listening to lectures in a traditional virtual classroom [10]. This system is based on three kinds of agents: teacher agents, student agents and remote proxy agents. Teacher agents interact with teachers and are responsible for: i) disseminating information to student agents and remote proxy agents, ii) maintaining student profiles and, on the basis of these profiles generating individual quizzes and exercises, iii) filtering students questions, and iv) managing classroom sessions progress. Student agents support the interaction with the teacher, maintain the profiles of the other students to identify potential "helpers" and, when it is necessary, solicits answers from such "helpers". Remote proxy agents support the interaction with the teacher and other students when a student is connected

with a low-speed internet connection (e.g., they filters messages to reduce the traffic). Guardia Agent is an agent-based system aimed at supporting students working on team projects [22]. This system is based on agents, one for each student, that autonomously monitor the progress of a group project, suggest new ways in which the students can act to improve the progress of the project (e.g., a new allocation of tasks), and enhance the communication between members of the group.

IV. Conclusions

In this paper, we present a system called RAP (Remote Assistant for Programmers) with the aim of supporting communities of students and programmers during shared and personal projects based on the use of the Java programming language. RAP associates a personal agent with each user and this agent maintains her/his profile and helps her/him to solve problems proposing information and answers extracted from some information repositories, proposing "experts" on these problems and then forwarding their responses.

RAP has similarities with WBT [8], I-MINDS [10] and, in particular, with the Expert Finder system [21]. In fact, both these three systems provide agents that recommend possible "helpers". However, none of them provides the integration of different sources of information (experts, answers archive and code documentation), and none of them integrates in the user profile information about user's day-to-day work products with information obtained from the answers the user provided to the other users of the system.

A first prototype of the RAP System is under development by using JADE [3],[9], a software framework to aid the realization of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [6]. A large part of the first system prototype has been completed and some tests have been already done. In particular the tests regarding the recommendation of experts have shown encouraging results.

The RAP system will be used in some practical courses on JADE by students of the partners of the "@lis Technology Net" project and by students and researchers, involved in the ANEMONE project [2], for cooperating in the realization of agent-based software. Moreover, the RAP system will be used as a service of the Collaborator system [5]. Collaborator is a system that provides a shared workspace supporting the activities of virtual teams through a set of services as, for example, chat and multimedia interaction, meeting scheduling and synchronous sharing of applications [4].

After the completion, experimentation of the first prototype, we plan to try to improve the quality of both document and expert recommendation by applying and then comparing the most considered recommendation techniques and, eventually, trying their integration.

References

[1] @LIS Technet Home Page (2003). Available from http://www.alis-technet.org.

[2] ANEMONE Home Page (2003). Available from http://aot.ce.unipr.it:8080/anemone.

[3] Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework.. Software Practice and Experience, 31, (2001) 103-128.

[4] Bergenti, B., Poggi, A., Somacher, M.: A Collaborative Platform for Fixed and Mobile Networks. Communications of the ACM, 45(11), (2002) 39-44.

[5] Bergenti, B., Poggi, A., Somacher, M., Costicoglou, S.: COLLABORATOR: A collaborative system for heterogeneous networks and devices. In. Proc. International Conference on Enterprise System (ICEIS03), Angers, France (2003) 447-480.

[6] FIPA Specifications (1996). Available from http://www.fipa.org.

[7] Hazeyama, A., Nakako, A., Nakajima, S., Osada, K.: Group Learning Support System for Software Engineering Education - Web-based Collaboration Support between the Teacher Side and the Student Groups. In Proc. Web Intelligence 2001, Maebashi City, Japan, (2001) 568-573.

[8] Ishikawa, T., Matsuda, H., Takase, H.: Agent Supported Collaborative Learning Using Community Web Software. In Proc. International Conference on Computers in Education, Auckland, New Zealand, (2002) 42-43.

[9] JADE Home Page (1998). Available from http://jade.tilab.com.

[10] Liu, X., Zhang, X. Soh, L., Al-Jaroodi, J., Jiang, H.: I-MINDS: An Application of Multiagent System Intelligence to On-line Education. In Proc. IEEE International Conference on Systems, Man & Cybernetics, Washington, D.C., (2003) 4864-4871.

[11] Mattox, D., Maybury, M. and Morey, D.: Enterprise Expert and Knowledge Discovery. The MITRE Corporation, McLean, VA, (2000). Available from http://www.mitre.org/support/papers/tech_papers99_00/maybury_enterprise/maybury_enterprise.pdf

[12] Maybury, M., D'Amore, R. and House, D.: Awareness of Organizational Expertise. The MITRE Corporation, MacLean, VA (2000). Available from http://www.mitre.org/support/papers/tech_papers99_00/maybury_awareness/maybury_awareness.pdf

[13] McDonald, D.W.: Evaluating expertise recommendations. In Proc. of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, Boulder, CO, (2001) 214-223.

[14] McDonald, D.W.: Recommending collaboration with social networks: a comparative evaluation. In Proc of the Conference on Human Factors in Computing Systems, Ft. Lauderdale, FL, (2003) 593-600.

[15] Mungunsukh, H., Cheng, Z.: An Agent Based Programming Language Learning Support System. In Proc. International Conference on Enterprise System (ICEIS02), Auckland, New Zealand, (2002) 148-152.

[16] Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine Learning, vol. 27, (1997) 313–331.

[17] Pazzani, M., Billsus, D.: Adaptive Web Site Agents. Autonomous Agents and Multi-Agent Systems, 5, (2002) 205–218.

[18] Resnick, P., Neophytos, I., Mitesh, S., Bergstrom, P., Riedl, J.: GroupLens: An open architecture for collaborative filtering of netnews. In Proc. Conference on Computer Supported Cooperative Work, Chapel Hill, (1994) 175-186.

[19] Salton, G.: Automatic Text Processing. (1989), Addison-Wesley.

[20] Sun Java Glossary (2004). Available from http://java.sun.com/docs/glossary.html.

[21] Vivacqua, A. and Lieberman, H.: Agents to Assist in Finding Help. in Proc. ACM Conference on Human Factors in Computing Systems (CHI 2000), San Francisco, CA, (2000) 65-72.

[22] Whatley J.: Software Agents for Supporting Student Team Project Work. In Proc. International Conference on Enterprise System (ICEIS04), Porto, Portugal, (2004) 190-196.