Ontology-Based Remote Collaboration for the Development of Software System

M. Mari, A. Poggi, P. Turci

Dipartimento di Ingegneria dell'Informazione, Università di Parma
Parco Area delle Scienze 181/A 43100 Parma - Italy
{mari, poggi, turci}@ce.unipr.it

Abstract. RAP (Remote Assistant for Programmers) is a Web and multi-agent based system to support remote students and programmers during common projects or activities based on the Java programming language. RAP helps users to solve problems proposing information extracted from dedicated repositories and forwarding answers received from other users, recommended as experts. Its peculiar characteristic is the integration of the agent technology with the semantic Web technology. In fact, in order to improve filtering and recommendation techniques, RAP takes advantage of an ontological approach to user and document profiling. A RAP system is not a closed system, instead it is based on a dynamic network of RAP platforms managing groups of geographically localized users and documents. Therefore, recommendations should take into account of the accessible experts and documents. At this purpose, RAP users and documents profile management subsystems provide a mechanism that dynamically adapts the relevance of each profile. An initial prototype of the RAP System is under development by using JADE.

1 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 catalogued: such information may only be accessed by asking the right people. The challenge of finding relevant information then reduces to finding the "expert". But on the other hand, people may easily get tired of receiving banal questions or different times the same question; therefore, who wishes help for solving a certain problem should look initially for documents related to the problem and then possibly look for a possible expert on the topic.
In this paper we present a multi-agent based system called RAP (Remote Assistant for Programmers) that integrates information and expert searching facilities for communities of student and researchers working on related projects or work and using Java programming language.
Its peculiar characteristic is the integration of the agent technology with the semantic Web technology. Recently, we have seen an explosion of interest in ontologies as

artefacts to represent human knowledge and as a critical component in several applications. Moreover the “marriage” between agents and ontologies seems to be the kind of technology that can significantly improve recommender systems.
On the one hand, ontologies enable agents to communicate in a semantic way, exchanging messages which convey information according to explicit domain ontologies. On the other hand, an ontological approach to user and document profiling can significantly improve the outcome of the matching process.
Clearly we are aware that it is a challenge problem, nevertheless the realizations of prototype systems can be of great help in order to raise awareness of the concrete issues and delineate possible solutions.
In the next section we discuss the related work. In the third section we illustrate the RAP system, its architecture and a description of its behaviour, highlighting the traits related to the semantic interoperability. The fourth section is dedicated to the profile management. In section five we describe the current state of the implementation of a prototype of the system and some preliminary evaluation results. The paper ends by outlining some considerations regarding future possible developments, and by drawing some conclusions around the results of the work done.

2 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.
Some of the most important proposed systems are applied to the recommendation of Web pages and therefore they are not specialized for computer programming documents, even if they usually allow the customization for different subjects. GroupLens is the first system that used collaborative filtering for document recommendation [17]. 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.
Adaptive Web Site Agent is an agent-based system for document recommendation [15]. This system works on the documents of a Web site and recommends documents to visitors using different criteria: user preferences for the subject area, similarity between documents, frequency of citation and frequency of access.
Several prior systems support expertise recommendations. Vivacqua and Lieberman [20] 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. 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. WBT (Web Based Teaching) is an agent based collaborative learning support system providing community Web services [11]. The system is centred 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. 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 [12]. 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: disseminating information to student agents and remote proxy agents; maintaining student profiles and, on the basis of these profiles generating individual quizzes and exercises; filtering students questions; 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, solicit 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).

3 Agents for Remote Collaboration

A key feature of our system is that it relies on an agent-based infrastructure which gives it significant advantages. Besides being an ideal mechanism for implementing complex systems, agent technology is well-suited to applications that are communications-centric, based on distributed computational and information systems, and requiring autonomous components readily adaptable to changes. Indeed, the agent-oriented paradigm provides both an appropriate level of abstraction in modelling distributed applications and a natural merging of object orientation and knowledge-based technologies that can facilitate the incorporation of reasoning, learning and high-level dialogue capabilities to realize intelligent and adaptive applications. Furthermore its integration with other key emerging technologies, such as semantic Web, may be the basis for the realization of successful and innovative agent-based systems supporting remote collaboration applications.
The first prototype of the RAP system has been realized by using JADE [9], which is considered the reference implementation of the FIPA [8] specifications and one of the most used and promising agent development framework.

3.1 Multi-Agent System Architecture

The system is based on seven different kinds of agents: Personal Assistants, Code

Documentation Managers, Answer Managers, User Profile Managers, Mail Manager, Starter Agents and Directory Facilitators.

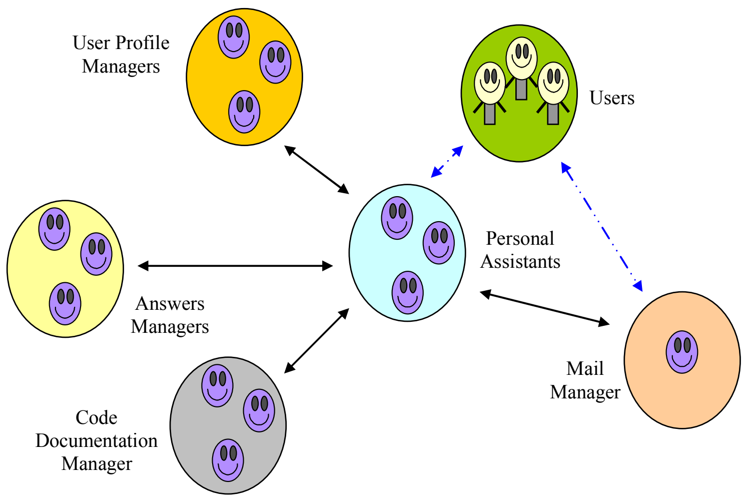

Fig. 1 - Main interactions among agents and users

Personal Assistants are the agents that allow the interaction between the user and the different parts of the system and, in particular, among the users themselves. This agent is also responsible for building the user profile and maintaining it when the user is "on-line". User-agent interactions can be performed in two different ways: when the user is active in the system, through a Web-based interface; when he/she is "off-line", through e-mails. Usually there is a Personal Assistant for each on-line user, but, when needed, Personal Assistants are created to interact with "off-line" users via e-mail.
User Profile Managers are responsible for maintaining the user profiles and for activating Personal Assistants when it is necessary that they interact with their "off-line" users via e-mail.
Code Documentation Managers are responsible for maintaining code documentation and for finding the appropriate "pieces of information" to answer the queries supplied by the users.
Answer Managers are responsible for maintaining the answers done by users during the life of the system and for finding the appropriate answers to new queries of the users. Besides providing an answer to a user, this agent is responsible for updating the score of the answer and forwarding the vote to either the Personal Assistant or the User Profile Manager for updating the user profile.
Mail Manager is responsible for receiving e-mails from "off-line" users and forwarding them to the corresponding Personal Assistants.
Starter Agent is responsible for activating a Personal Assistant when either a user logs on or another agent requests it.
Directory Facilitator is responsible for informing an agent about the addresses of the other agents active in the system (e.g., a Personal Assistant can ask about the address of all the other Personal Assistants, the Code Documentation Managers, etc.).

Fig. 1 gives a graphical representation of a RAP platform focusing on the major interactions. 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. Finally, in a RAP system there is a Directory Facilitator for each platform.

3.2 System Behaviour

A quite complete description of the system behaviour can be given showing the scenario in which a user asks information to her/his Personal Assistant to solve a problem in her/his code and the Personal Assistant finds one (or more) "pieces of information" that may help her/him. The description of this scenario can be divided in the following steps: (1) selecting answer types, (2) submitting a query, (3) finding answers and (4) rating answer.

(1) Selecting 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.

(2) Submitting a query: the user, through the interface, provides the query to her/his Personal Assistant. 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 contain concepts an individuals from the domain ontology 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 content of the query.

(3) Finding answers: the Personal Assistant performs different actions and interacts with different agents to collect the various types of answers. For getting code documentation, the Personal Assistant asks the Directory Facilitator about all the Code Documentation Managers. The Personal Assistant forwards the query to all these agents; these agents search "pieces" of code documentation related to the query and send them to the Personal Assistant associating a score with each "piece". For getting answers from the answer system repositories, the Personal Assistant asks the Directory Facilitator about all the Answer Managers. These agents search answers related to the query and send them to the querying Personal Assistant 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: (3.1) finding experts, (3.2) receiving experts rating, (3.3) selecting experts, (3.4) receiving answers.

(3.1) Finding experts: the Personal Assistant asks the Directory Facilitator about the other active Personal Assistants (e.g., the Personal Assistants of the user that are "on-line") and all the User Profile Managers of the system (e.g., the agents managing the profile of the users that are not "on-line"). After receiving this information, the Personal Assistant forwards the query to all these agents.

(3.2) Receiving expert rating: Personal Assistants and User Profile Managers com-

pute the rating of their users to answer to the query on the basis of the query itself and of the user profile. Agents that compute a positive score (e.g., their users may give an appropriate answer to the query) reply to the querying Personal Assistant with the rating of a single user (in the case of a Personal Assistant) or a certain number of users (in the case of User Profile Manager).
(3.3) Selecting experts: the Personal Assistant divides on-line and off-line users, orders them on the basis of their rating and, finally, presents these two lists to its user. Then, the Personal Assistant sends the query to the selected Personal Assistants or User Profile Managers.
(3.4) Receiving answers: the selected Personal Assistants immediately present the query to their users and forward the answer as soon as the users provide it. User Profile Manager activates the Personal Assistants of the involved users through the Starter Agent; these Personal Assistants forward the query to their users via e-mail and then terminate themselves. Users can answer either via e-mail or when they log again on the system. In the case of e-mail, the Mail Manager starts the right Personal Assistant that extracts the answer from the e-mail and forwards it. When the querying Personal Assistant receives an answer, it quickly forwards it to its user.
(4) Rating answers: after the reception of all the answers, or when the deadline for sending them expired, or, finally, when the user has already found an answer satisfying her/his request, the Personal Assistant presents the list of the answers to its user asking her/him to rate them. Afterwards, the agent forwards each rating to the corresponding Personal Assistant, Code Documentation Manager, Answer Manager or User Profile Manager 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.

3.3 Agent Communication

A major aim of multi-agent systems is to enable software integration on a deeper level, namely shifting the integration process from syntactic interoperability to semantic interoperability.
During the last years, the FIPA standard organization produced a comprehensive set of specifications for interoperable multi-agent systems. It defines the agent communication language, the requirements for a content language, the technologies enabling agents to manage ontologies, containing entities from the domain of discourse.
In Fig. 2 the FIPA Communication Conceptual Model is represented. Two semantic levels can be distinguished. The semantics at communication protocol level, which is domain independent, and the semantics at content language level, which most likely depends on the application domain. The information is represented as a content expression, consistent with a proper content language and with an application-specific ontology, defining concepts and their relations.
From the figure it is clear that the reference ontology is used as a “contract” between agents undertaking a conversation; in other words they each claim to be using an interpretation of the terms, used in the content of messages, conforming to the specified

ontology.
Though ACL semantics and a shared ontology are essential, they still are not enough to achieve semantic interoperability: the conversational and social contexts need to be taken into account. The FIPA standard, therefore, contains a set of interaction protocols, detailing standard conversation templates that can be reused.

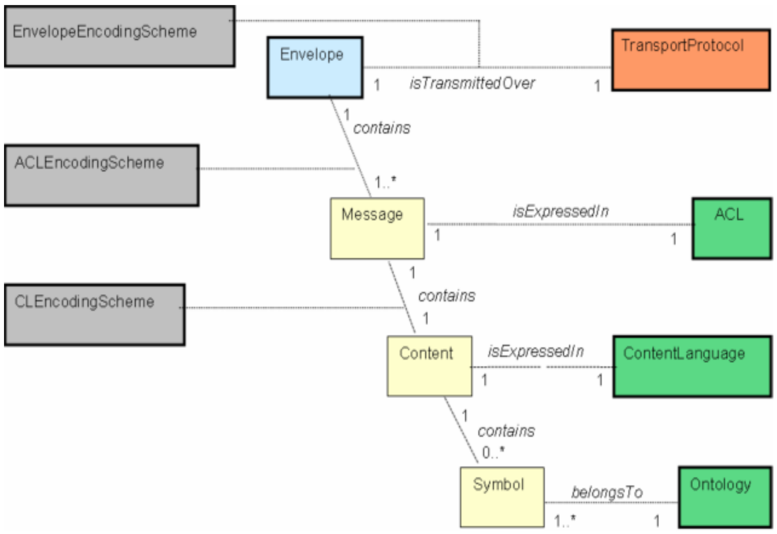

Fig. 2 - FIPA Communication Conceptual Model

In the following subsections the issues connected with the ontological support and the formal language and protocol for querying agents, both required in order to support and facilitate a querying-answering dialogue among agents, are dealt with in more details, with a particular focus on the agent platform used (i.e. JADE).

3.3.1 Ontological Support

JADE offers a general support for ontologies with the goal of enabling agents to communicate in a semantic way, exchanging messages which convey information according to explicit ontologies. The idea which mostly inspired its design was to define an ontology independent abstract model of the content language that could be subsequently bound to any domain ontology representation expressed using an object-oriented data model. The model of the content language is FIPA compliant and is able to describe: i) concepts, construct that represents an identifiable entity; this is mainly important to realize a typed knowledge base; ii) predicates, identifying the status of a part of the world; iii) actions, expressing the actions that can be performed by the agents, and iv) identifying reference expressions, identifying the entities for which a given predicate is true.
This ontological support has been conceived when the Semantic Web was on its very early stage of research and development and OWL was not already established as a standard. Consequently its expressive power is clearly limited with respect to OWL and basically allows expressing taxonomy of concepts, predicate and actions and

therefore it is not able to represent completely the different application domains where JADE agent may be used.
In order to provide a JADE agent with an adequate expressive power (i.e., equivalent to the one offered by OWL DL [3]), it is necessary either to replace or to integrate the JADE ontological support. In the attempt of finding a suitable solution to this problem one has to cope with a scenario characterized by different domain knowledge modelling techniques and by different needs. While on the web the increasing interest in ontologies is driven by the large volumes of information available and by the need of automating many information retrieval activities that were traditionally performed manually, in the agent context the focal point is slightly different and it is mainly on communicative acts - communications which implies actions. In particular a peculiar characteristic of the agent community is the heterogeneity of resources available and the roles played by different agents of a system. This leads us to choose different approaches in different contexts. Our solution was to realize a compound tool, called OWLBeans [6], that allows the use of ontologies described by using OWL DL. These ontologies can be used by agents both for performing their tasks in cooperation with other agents and for interoperating with external entities (e.g., legacy software systems). OWLBeans is based on a two-level approach with the aim of coping with both the issues of managing complex ontologies and of providing ontology management support to lightweight agents, which seldom need to deal with the whole complexity of a OWL DL ontology. Therefore, lightweight agents maintain the simple JADE ontology support whereas one or more dedicated agents, acting as ontology servers, are able to use and manage complete OWL DL ontologies and provide service to the agents that need it.
The main functionality of OWLBeans is to extract JADE ontologies from OWL DL ontologies realizing a set of ontologies usable by JADE agents, with the obvious shortcoming that not all the information maintained in the original OWL ontologies are taken into account. Therefore, for all those systems that need a complete support for OWL DL ontologies, OWLBeans offers a set of ontology server agents, implemented as JADE agents, providing a common knowledge base and reasoning facilities. These ontology servers use the Jena toolkit [10] to load, maintain and reasoning about OWL ontologies. The other agents of the system do not need to know anything about the Jena toolkit given that these ontology servers provide them with a set of simple actions for querying and manipulating the ontologies. Furthermore OWLBeans takes into account proper authorization mechanisms. In particular, the underlying JADE security support has been leveraged to implement a certificate-based access control. Only authenticated and authorized users will be granted access to managed ontologies. The delegation mechanisms of JADE allow the creation of communities of trusted users, which can share a common ontology, centrally managed by the Ontology Server.
Finally, despite the fact that the JADE ontological support is quite simple, it could still be complex for some devices with limited resources such as smart phones. This is the reason why we have decided to improve OWLBeans adding a further feature which allows agents to import taxonomies and classifications from OWL ontologies, in the form of a hierarchy of Java classes with the purpose of providing very simple artefacts to access structured information. Given its modular architecture, based on an interme-

diate ontology model, OWLBeans also provides further functionalities, e.g., saving a JADE ontology into an OWL file, or generating a package of JavaBeans from the description provided by a JADE ontology.

3.3.2 OWL-QL as a Query Language and Protocol

The messages exchanged by JADE agents have a format specified by the ACL language defined by the FIPA international standard for agent interoperability. This format comprises a number of fields, which are almost all optional. The only one that is mandatory in all ACL messages is the performative, indicating what the sender intends to achieve by sending the message. In particular the "query-if "performative can be used when the sender wants to know whether or not a given condition holds while the "query-ref" is the act of asking another agent to inform the requester of the object identified by a descriptor. The message content of these two performatives is respectively a proposition and a referential expression. FIPA does not indicate any content language as mandatory even if some languages are proposed; among these the SL language.

Agents can be involved in conversations, that is sequences of messages exchanged by two or more agents with well defined causal and temporal relations. Complex conversations are typically carried out following a well defined interaction protocol.

Considering the RAP system, we have decided to use the support provided by JADE for all the conversations, except for the specific conversations concerning data retrieval. After a in depth analysis of the possible dialogues which can happen between an agent asking for information and an agent providing it, we have come to the decision of utilizing the semantic Web technology, even if in this field it is still in its infancy.

The reason of our choice was twofold: (i) the content languages provided by JADE for querying information, that is the SL language, is not widely used and quite complex; (ii) the query interaction protocol, the only suitable for information retrieval, is quite limited since it does not support a conversation between a server and a client consisting of several steps. Indeed, in general we cannot expect that a server will produce all the answers at once, moreover the client normally prefers to receive the results of an incomplete search instead of waiting an exhaustive search to be completed.

For this reasons we have decide to use a query language targeting the Ontology Web Language and specifically OWL-QL [7], a formal language and protocol for query-answering dialogues among semantic-aware agents using knowledge represented in OWL.

4 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.

In order to simplify, speed up and reduce the possibility of inaccuracy, we decided to build the initial profile of the users and documents in an automated way which, for the users, is very similar to the one used by Expert Finder system [20]. 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 the terms included in the domain ontology, provided to the users for annotating their queries, and to the names of the classes and methods of the Java and JADE software libraries used by the community of the users of the system.

Document and user profiles are computed by using “Term Frequency Inverse Document Frequency (TF-IDF)” algorithm [18] and profiles weighted terms correspond to the TF-IDF weights. Some problems have risen applying this approach in our multi-platform and distributed system: we present these problems and discuss the solutions in the following subsection. Each user profile is built by user’s Personal Assistant through the analysis of the Java code she/he has written. The profile built by Personal Assistants is only the initial user’s profile and it will be updated during the system lifetime 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: (1) a user asks about a problem and then rates some of the received answers, (2) a new software library is added to the ones used by the community or some new terms are introduced in the domain ontology, and (3) a user writes new software.

In the first case, there are three possible behaviours according to the source of the answer (user, document repository or answer repository). If the answer comes from a user, her/his profile is updated on the basis of the expressed rating. 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 related rating. If the answer comes from the document repository and the rating is positive, the answer is added to the answer repository and its profile is the original document profile updated by the new rating. 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, 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 (e.g., the answer was good for one or more previous queries, but also for the current one).

We decided to avoid the use of negative evaluation 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 the answer useful to solve her/his problem; therefore, if later this answer received a negative score, it only means that the answer is not appropriate for the last query.

When a new software library is available for the users of the system or some new terms are introduced in the domain ontology, all the document and user profiles have to be updated. While document profiles are rebuilt on the basis of the new complete set of terms, user profiles are updated adding the new weighted terms, of course with a weight equal to their frequency in the user’s Java code.

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

4.1 Open and Distributed Communities

An important requirement that has guided the design of RAP has been the support for open and distributed user communities. In fact, the retrieving of experts and information can take a great advantage if the community has the capability to grow and include new users or new sub-communities.

The community beneath RAP is distributed and the whole system can consist of dynamic groups of local communities. Each community can exist and operate isolated, but can also decide to join a group of communities, sharing the experts and the document repositories. The joining and the leaving of a community are dynamic operations; the single communities of a group are fully independent, just like the components of a peer-to-peer network.

The open and distributed nature of the system provides the best conditions for sharing and retrieving information, but also entails some significant problems in profile evaluation. In fact, the evaluation of both experts and documents strongly depends on the actual composition of the community group. For instance, if at a given moment a user is rated the maximum expert to answer a particular query, he is rated considering only the users registered in the system at that moment. If a new local community joins the group, it is possible that a user with more experience has become available. In this case, information like "user A has called n times a method of the class X" is still valid, but information such as "user A is the maximum expert of class X" may change according to new composition of the community.

The problem rises from the fact that, while the user personal information is still valid, the rating and all other information related to the community should be recalculated. 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.

Only the Personal Assistant can access, for privacy and security reasons, to the software of its user and the profiles are maintained by the corresponding Personal Assistants or by User Profile Managers when the Personal Assistant is not alive. For these reasons, each profile component of the RAP system is associated with two elements: an absolute element and a TF-IDF weighted element. The absolute one depends only on the user (or document) profile and it is updated only in the cases described in the previous section; the TF-IDF weighted element depends on both the user profile and the whole community profiles. While the absolute element is stored in a database, the weighted one is maintained in memory and it is recalculated when necessary. Obviously, every rating is determined on the basis of the weighted elements.

The situations in which it could be necessary to recompute the weighted element of the profile components can be slightly frequent:

- a new community joins or leaves the community group;
- a new user registers or is deleted from the system;
- some components of a user profile change: for example the user submits new software or receives a rating for an answer.

4.2 Ontology-Based Profiles

The Rap system is basically characterized by two kinds of models (a code model and a domain model, which in turn originates from the integration of two domain models: Java programming language and JADE framework) and two kinds of profiles (user and answer/document profiles) based on the ontologies corresponding to the models.
As far as our specific domain model and the code model are concerned the research is still at the beginning and the corresponding ontologies are almost missing. Therefore the ontologies built in the context of RAP system are still just experimental. The methodology we have adopted in ontology engineering has been based on two essential principles. First, we would like our ontologies to gradually evolve. We have started with largely incomplete ontologies, which will most likely grow incrementally and iteratively over time. Second, we have exploited state-of-the-art languages and tools provided by W3C Consortium.
Using ontologies instead of a flat list of unstructured topics, such as the SUN "Glossary of Java Related Terms" [19], has different advantages.
First of all the results of the queries can be more precise if relationships such as part-of, prerequisite and is-a are considered, since such relationships increase the accuracy of profiles and hence the usefulness of recommendations. Secondly the number of keywords, selected among the terms that are frequently referred in documents, can be very high with respect to relevant ontological concepts required to define an exhaustive domain ontology. Therefore the use of ontologies would decrease the profile size, without loss of information, requiring a smaller amount of time for reasoning. Moreover we would have a more flexible way of classification and different levels of detail from the hierarchy could be used to represent profiles. Lastly reasoning can be employed to augment the data and automate the localisation of even more information, allowing expertise and interests to be discovered that were not directly observed for instance in the user's behaviour.
Choosing to follow the ontology approach, the first problem decision we had to make was to settle on the relations to be used to map the list of keywords to the concepts of the ontologies. In recent years the relation between ontological knowledge and lexical resources has been investigated by researchers. In [16] an analysis of the different methods which may guide the linking of ontologies and lexical resources have been carried out, leading to three possible methodological options. In our case we have estimated that a satisfactory approach would have been to define an ontology (basically the TBox part only) and populating it with lexical information.
Our domain model is a description of the terms of required know-how. We have implemented a very simple version of an ontology whose concepts were inspired by ACM2002 classification hierarchy; namely "area", "unit", "topic" and "subtopic". Afterwards we have populated the ontology with individuals picked out among the terms belonging to the Sun "Glossary of Java Related Terms", the computer science classification made by the Open Directory Project [14] and the JADE glossary. We have chosen to refer to existing glossaries and taxonomies in order to speed up the building time and provide a potential route for the integration with other future ontologies. We have initially identified three primary relationships, namely is-a, part-of and prerequisite relationships. Is-a relations are used to indicate specialization of

concepts while part-of and prerequisite relations denote required sub-concepts for the understanding of a given concept. In this way we are able to set up an initial ontology describing our view on know-how about our application domain.
The code model on the other hand was implemented with a simple ontology which was derived empirically on the basis of the kind of searches performed by the developers (e.g. packages, classes, methods, etc.). The part of this ontology concerning the individuals is generated automatically from the code. The code manager agent reads the code, extracts the needed information and finally creates an OWL file which will be processed by the Jena toolkit embedded in the manager agents. The ontology representing the code model contains very few concepts but it is populated with a large number of individuals.

5 System Implementation and Evaluation

A first prototype of the RAP System is under development by using JADE [5]. 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. JADE is an open source project, and the complete system can be downloaded from JADE home page.
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 it is connected to a Web server, for allowing direct interactions with the users, and to a mail server, for allowing e-mail interactions with the users. In each agent platform there is a unique Starter Agent and Mail Agent, but there might be more than one User Profile Manager, Code Documentation Manager and 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 too (in this case, agents manage copies of data). 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 significant part of the first prototype of the system has been completed. In particular, the subsystem supporting interactions among Personal Assistants and the interaction between each pair of Personal Assistant and the "on-line" user has been completed. This subsystem has been used with success by a small set of students, connected from different labs or from home, for the development of JADE software within some course works. In these activities, students could formulate queries annotating them with concepts or class and method names (from source code) extracted from both domain and code ontologies.
Even if in a very early stage of development we have been able to determine the effect of the depth of the reference ontology on the accuracy of the profile. We have found

that profiles consisting of only the first two levels of ontology (namely, area and unit concepts) provide essentially no personalization, whereas expanding the profile from two levels to three and so on, i.e. moving towards more specific concepts, has allowed us to build a more detailed profile with only a small drop in precision. Finally we have found that a four-level ontology (i.e. area, unit, topic and subtopic) is a good compromise between profiles precision and the time needed to compute them. Moreover, we have performed some tests to compare the use of the system when the users perform questions by using the keywords belonging to a glossary and when they also use the ontological support. The result of these tests is that the use of the ontological support reduces the number of queries necessary to get the right information of about a 20% (of course, the right value of the query reduction will be given by a more exhaustive and rigorous set of tests).

6 Future Work

The evaluation results are quite promising, but they have shown a main area of possible improvement. The quality of the results is indeed greatly affected by the accuracy of the user and document profiles. Our research work resolves around using ontologies to represent users and documents conceptually. Due to the lack of ontologies for our specific domain and the absence of powerful tools to support the building of domain ontologies, the ontologies used in the prototype represent a first attempt of classification ad are essentially experimental. Since our background was mainly in the field of conceptual modelling and so we were moderately novices in ontological modelling, our first choice was directed towards a straightforward approach, that is we decided to map lexical units to ontological concepts, focusing basically on the “individual level”. Using a most complete approach we could build more rigorous and effective ontologies from both the point of view of the number of concepts and the relationships between them. This would allow more powerful inference and thus give a significant boost to profiling accuracy.
To this purpose we are developing a new ontological support with the expressive power of OWL DL but more efficient from the point of view of memory and time consuming than Jena. In particular, we are developing a set of Java based software tools to handle, maintain and reasoning about OWL ontologies. These software tools use an object-oriented model of OWL ontologies, more concise than the one proposed in OWL-API [4], allowing a complete representation of OWL DL ontologies.
Considering the user profile there are several working groups and efforts with the aim of defining a standard vocabulary. Since the model of a RAP user is quite close to that of a learner in order to improve the accuracy of the matching process we will enrich the RAP user profile with reference to the two efforts of standardizing learner profile, that is IEEE Learning Technology Standards Committee and IMS Global Learning Consortium. These standards reflect different perspectives. The structure of the IMS Learner Information Package standard was derived from best practices in writing CV’s and inter-personal relationships are not considered at all. The IEEE PAPI standard on the other hand has been developed from the perspective of a learner perform-

ance during his study. The main categories are thus performance, portfolio, certificates and relations to other people. Our present user profile model will be improved adding a subsets of both mentioned standards enhanced with some specific extensions for RAP system. This will allow us to take into account new traits, such as behavioural aspects (e.g. the level of activity of the users in the system) or the user's role in the company, that will contribute in defining a more accurate and valuable profile.

7 Concluding Remarks

In this paper, we have presented 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 Java programming language. RAP associates a Personal Assistant 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 [11], I-MINDS [12] and, in particular, with the Expert Finder system [20]. In fact, all 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 has provided to the other users of the system. Another original contribution of RAP is the design of a recommendation system composed by an open and distributed group of communities. Each community is independent and can dynamically join or leave a group. The system automatically updates all the profile components that may vary on the basis of the group composition.
A first prototype of RAP is under development: a part of the system has been completed and some tests have been already done. In particular, the tests regarding the recommendation of experts have shown encouraging results.

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. Bechhofer, S., van Harmelen, F., Hendler, J. Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., & Stein, L.A. OWL Web Ontology Language Reference. 2004. Available from http://www.w3.org/TR/owl-ref/.
4. Bechhofer, R. Volz, and P. Lord. Cooking the semantic web with the OWL API. In Proc. Int Semantic Web Conference, pages 659 - 675, Sanibel Island, FL, 2003.
5. Bellifemine, F., Poggi, A., Rimassa, G. Developing multi-agent systems with a FIPA-compliant agent framework. Software Practice and Experience, 31 (2001), 103-128.

6. Bergenti, F., Poggi, A., Tomaiuolo, M., Turci, P. An Ontology Support for Semantic Aware Agents. In Proc. Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005 @ AAMAS), Utrecht, The Netherlands, 2005.
7. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL - A Language for Deductive Query Answering on the Semantic Web. Technical Report KSL-03-14, (Knowledge Systems Lab, Stanford University, CA, USA)
8. FIPA Specifications (1996). Available from http://www.fipa.org.
9. JADE home page (1998). Available from http://jade.tilab.com.
10. Jena, HP Labs Semantic Web Toolkit software and documentation. http://jena.sourceforge.net/
11. Ishikawa, T., Matsuda, H., Takase, H. Agent Supported Collaborative Learning Using Community Web Software. Proc. International Conference on Computers in Education, Auckland, New Zealand, 2002, 42-43.
12. Liu, X., Zhang, X. Soh, L., Al-Jaroodi, J., Jiang, H. I-MINDS: An Application of Multi-agent System Intelligence to On-line Education. Proc. IEEE Int. Conference on Systems, Man & Cybernetics, Washington, DC, 2003, 4864-4871.
13. McDonald, D.W. Evaluating expertise recommendations. Proc. of the 2001 Int. ACM SIGGROUP Conference on Supporting Group Work, Boulder, CO, 2001, 214-223.
14. Open Directory Project, home page http://dmoz.org/
15. Pazzani, M., Billsus, D. Adaptive Web Site Agents. Autonomous Agents and Multi-Agent Systems, 5 (2002), 205–218.
16. Prevot, L., Borgo, S., Oltremari, O.: Interfacing Ontologies and Lexical Resources. Proc. of the Ontologies and Lexical Resources: IJCNLP-05 workshop 2005
17. Resnick, P., Neophytos, I., Mitesh, S., Bergstrom, P., Riedl, J. GroupLens: An open architecture for collaborative filtering of netnews. Proc. Conference on Computer Supported Cooperative Work, Chapel Hill, 1994, 175-186.
18. Salton, G. Automatic Text Processing. Addison-Wesley, 1989.
19. Sun Java Glossary (2004). Available from http://java.sun.com/docs/glossary.html.
20. Vivacqua, A. and Lieberman, H. Agents to Assist in Finding Help. Proc. ACM Conference on Human Factors in Computing Systems (CHI 2000), San Francisco, CA, 2000, 65-72