A multi-agent system for acquiring and sharing lessons learned

Cesar A. Tacla*, Jean-Paul Barthès

CNRS UMR 6599 HEUDYSIC, Computer Science Department, University of Technology of Compiègne, BP 20529-60205 Compiègne Cedex, France

Abstract

This paper presents a multi-agent system for knowledge management (KM) in research and development (R&D) projects. R&D teams have no time to organize project information, nor to articulate the rationale behind the actions that generated the information. Our aim is to provide a system for helping team members to explicit knowledge, and to allow them to share their experiences, i.e. lessons learned (LL), without asking them too much extra-work. The article focuses on how we intend to help the team members to feed the system with LL, using the operations they perform on desktop computers, and how we intend to exploit the LL by using a case-based reasoning engine. We have been developing a prototype of such a KM system for a cooperative project.

Keywords: Multi-agents; Knowledge management; Architecture; Lessons learned

1. Introduction

This paper concerns knowledge management (KM) in research and development (R&D) projects. R&D team members have no time to organize documentation nor to articulate the rationale behind the set of decisions and actions found in a project. At the end of the project, the group has a naturally distributed memory as each member played a different role and consequently has a different view of common experiences. Thus, our overall goal is to build a system for supporting collaborative work, organizing project documentation and acquiring lessons learned (LL) without overloading the team members with extra-work. We use the terms experience and lessons learned as synonyms as they both refer to something a team member has learned or experienced during the life of a project (e.g. a solution for a problem, or the justifications for a project decision).

According to Shipman and McCall [1], documentation has a narrower scope than experiences, given that documentation generally contains only immediate explanations of the results. Rejected alternatives, pursued dead-ends and basic philosophical discussions are not relevant to documentation although they are an integral part of experience. Hence, we need to provide some mean for preserving and exploiting experiences to the project members rather than simply recording their documents.

R&D projects are fertile fields for the team members to learn new lessons since they can find knowledge intensive tasks often not completely prescribed in the model of tasks of the project. Such tasks are weakly structured, that is, they have neither a well-defined sequence of sub-tasks, nor a known collection of required resources. For example, consider a team member writing a prospective report who lacks information concerning a certain subject. She gives herself...
the task “get information about the subject x” and performs the following operations: she searches for documents on the Web, starts a chat session, and posts questions in a discussion forum. When she obtains enough information for completing the report, she estimates that the task is finished. The operations can be used as the basis for an LL record, containing a problem description (e.g. obtaining information about the subject x), a context description (e.g. the major project task in progress and the people involved), and the obtained resources (e.g. the documents, the specialists). This article focuses on how we intend to automatically capture the operations performed on a desktop computer by using personal assistant agents, and how we intend to assist the users to organize them as LLs. We are also interested in how agents help users in the sharing of the acquired LLs.

Section 2 presents the guidelines for the KM system including the principles, the reasons for using a multi-agent architecture, and the functional requirements for the system. In Section 3 we introduce our multi-agent platform that will enable to discuss in Section 4 the KM system, meaning, the kind of agents and their arrangement as a team. Section 5 gives an example for clarifying how the KM system works in practice, presents more details on the capture of operations and on the process of acquiring and reusing experience. In Section 6, we present related works following the current development status of the system, and finally we offer a conclusion.

2. Guidelines

In this section, we first present the principles that have guided us in some architectural decisions. Then, we point out the reasons for using a multi-agent architecture, and finally we present the functional requirements for the KM system.

2.1. Principles

We have adopted some principles that have guided us in our decisions concerning the architecture for the KM system. The very underlying principle is the concept of *Ba* proposed by Nonaka and Konno [2]. Very briefly, a *Ba* may be considered as a framework for managing and creating knowledge.

The second principle is practical. In order to offer possible solutions for KM systems, one has to take into account the reality of the situation, and in particular organizational constraints (e.g. team members available time, or the presence of legacy systems), rather than devising elegant but impractical systems.

The third principle is to consider knowledge as a personal item that can be developed individually and maintained locally. Knowledge may be expressed in the information produced, retrieved and exchanged among members of the project. Thus, we try to organize the information automatically and locally in a per user mode. The result is a distributed group memory where each member keeps part of the knowledge of the team.

2.2. Selected technology

To develop our KM environment we selected the multi-agent technology. The first reason to employ multi-agents in KM systems is that, like in a team, a multi-agent system is composed of a group of possibly heterogeneous and autonomous agents that share a common goal and work cooperatively to achieve it. A multi-agent system can display an intelligent behavior. For example, using techniques from artificial intelligence, agents can maintain a representation of what is happening and offer spontaneous help (proactive behavior). To do so, they can use past information, for example, through case-based reasoning techniques. In addition, since agents are developed independently, a multi-agent system can be easily modified by simple addition or removal of agents like in a team where members enter and quit the group at any time.

The second reason is that most of the currently available commercial solutions for KM systems are based on CSCW/groupware tools. Such tools provide the foundation for KM but we claim that they do not support adequately the process of knowledge creation. For example, the way the users perform their regular activities is not effectively taken into account by groupware at present time, though this may be an important source of knowledge to be reused. Other researchers have remarked that CSCW and KM tools, in general, are not very well integrated, meaning that KM is sometimes considered as an isolated activity [3]. Lewkowisky and Zachlad [4] state that groupware should be augmented to support structured dialogues...
for expressing the design rationale. In fact, this argument has been employed in several projects with the purpose of trying to capture the design rationale while the design is being realized (e.g. [1,5,6]). The point here is two-fold: KM systems must aid users to express or represent two types of knowledge, i.e. tacit (the design rationale, the experiences) and explicit (e.g. reports, papers, manuals), and for this the KM systems must be integrated to day-to-day activities to avoid increasing the team members regular work load.

We think agents are better equipped for learning from monitoring the users’ actions than groupware. An agent may have mechanisms for learning the user’s preferences and the context of the work allowing it to suggest some action or to propose a service. Hence, agents are autonomous and can act pro-actively, two important features for supporting users in exploiting existing knowledge and for helping to build an organizational memory.

2.3. Functional requirements

The KM system we propose must be able to provide the following functions: (i) automatically organize documents, and support the team members in retrieving and sharing them; (ii) aid team members to articulate their tacit knowledge or to share formalized knowledge. We use the concept of lessons learned as a means to represent formalized knowledge, although other representations could be used like decision trees to represent the decisions taken in a project; and (iii) support the collaborative work by managing the team tasks, allowing for asynchronous and synchronous communication among team members, and providing a persistent and reliable environment.

In this paper, we focus mainly on the first two requirements, given that groupware tools mostly provide the third one. The next section introduces the multi-agent platform that was used. The presentation of the platform will enable the discussion about a KM system (Section 4) satisfying the first two requirements.

3. OMAS: a multi-agent platform

In our laboratory we have been developing agent systems for a long time, in particular in the domain of complex engineering design. Monceyron and Barthès [7] developed a blackboard design environment, EXPORT, applied to harbor design. More recently Shen and Barthès [8] developed a generic multi-agent platform, DIDE, for mechanical engineering design. Thereafter, Scalabrin et al. [9] developed a generic platform for cognitive agents. These efforts led to the development of the open multi-agent system (OMAS) platform whose main features are summarized in the following paragraphs.

3.1. Cognitive agents

OMAS hosts cognitive agents. Such agents contain “an explicitly represented, symbolic model of the world and in which decisions are made via symbolic reasoning” [10]. The main advantage of cognitive agents is the possibility of designing intelligent behaviors by means of a set of skills. Moreover, such agents are autonomous, running independently of any particular task. It allows them in particular to display a proactive behavior, which makes them very different from client-server systems. A large number of papers and books have been published about agents and the reader is referred to Jennings et al. [11] for a quick review of the domain.

The OMAS platform provides two types of agents.

(a) Service agents (SAs) that provide a particular type of services corresponding to specific skills (one of the services is the interface to foreign platforms).

(b) Personal assistants (PAs) that are in charge of interfacing humans to the system. Their particular skills are devoted to understanding their masters and presenting the information in a timely fashion.

The internal structure of an agent is given in Fig. 1. It can be seen that a PA is an SA augmented with a user interface and a model of the user (the gray area). Briefly stated the components of an agent are the following.

- **Net interface**: implements the communication protocols.
- **Control module**: is responsible for the agent behavior.
- **Skills**: contain the set of services of the agent. A particular skill can be expressed as procedures or
rules. If the skill is complex, then a plan for executing it can be created, and the derived tasks are spawned and broadcast on the network.

- **World**: contains a representation of the other agents (their skills) and of the environment.
- **Tasks**: a task is a representation of what the agent is currently executing.
- **Ontology**: deals with agent ontology and domain ontologies. The agent ontology is employed to represent the agent skills, the tasks, and the skills of the other agents.
- **Self**: contains the representation of the agent’s skills, data stored in its memory, and possibly its own goals.
- **User model**: contains user preferences and a model of the dialog with the user.
- **User interface**: allows for dialoging with the user.

### 3.2. Communication

In our platform the agents are tightly linked within groups called coteries. Each agent inside a coterie receives all messages, allowing it to update and maintain its own internal representations, like a person in a small group hears all conversations within a certain distance. A coterie however can participate in higher-level interactions, that is, coteries can be linked by means of broker agents (our broker agent is called transfer agent). Inside a coterie there is not a control structure, or a concept of group or of social structure. There can be however an organization structure among people involved in the coterie (i.e. people using the agents). The inspiration for the concept of coterie comes directly from the notion of Ba [2].

Communication within a coterie occurs asynchronously in broadcast mode (messages can be directed, e.g. point to point, but communication mode is broadcast). This probably will appear surprising to some readers and interpreted as a waste of bandwidth. In practice, since the agents of a coterie share the same local network, each agent listens to all the messages at the lowest level. As messages are usually filtered out in a point-to-point communication, broadcast, implemented at a low level (e.g. by means of a UDP protocol), is essentially free.

At a higher level, we must make use of some communication protocols. For processing tasks, we extended the Contract-Net protocol [12]. The resulting version, the B-Contract-Net, allows for the agents to run at two levels of priority and to work on contracts even if they were not selected. The B-Contract-Net is quite complex and out of the scope of this paper. Note however, that although there are a number of exchanges during the first phase of Contract-Net type protocols, we do not consider such protocols as negotiation protocols, but as a coordination method for task allocation, agreeing with Jennings et al. [11].

In relation to the content language, ontologies play an essential role. Domain ontologies are used as primitives of the agent content language, assuming a KQML-like message structure [13] and a content language resembling SL0, SL1 or KIF [14]. SAs use ontologies to interpret the incoming messages and thus to provide the required service. Ontologies are also used by the SAs for building and modifying local representations. PAs employ the ontologies in the same fashion, but also for communicating with the user.

An interesting point of our platform is that it does not contain a directory service as recommended by the FIPA standard. A directory service is supposed to provide a location where agents register their descriptions composed of name, locator, and eventually other descriptive attributes characterizing the services.
offered by the agent. Each OMAS agent builds its own representation of the other agents from the messages that circulate in the coterie.

A new agent, wanting to advertise its services to the other agents within a coterie may either broadcast a message or simply answer corresponding calls for bids since they are all broadcast within a coterie. An agent from another platform wishing to communicate with an agent inside the coterie must send messages to a special SA, the transfer agent. This agent translates the received messages and broadcasts them to all the other agents in the coterie. Conversely, an agent inside the coterie must send messages to the transfer agent to interact with an agent outside the coterie on a different platform. Hence, the transfer agent provides facilities for communicating with other coteries or external FIPA compliant platforms.

4. The KM system architecture

This section describes the KM system architecture built using the OMAS platform. We point out the agents forming our architecture and how they are arranged into a coterie. We recall that we focus on the following functional requirements: document organization and experience acquisition and sharing.

4.1. The agents

The KM contains several types of agents (Fig. 2): personal assistants, organizers, project agents, transfer agents, repository agents.

The PAs play a major role in the KM system. Firstly, they are in charge of all exchanges of information among team members. For example, if a team member wishes to send an e-mail to a colleague, he asks for his PA that knows which SA is capable of effectively sending the e-mail. Secondly, the PA is able to organize the documentation of its master always with some help from the SAs. Finally, as R&D members have to deal with knowledge intensive tasks, they are supposed to construct their own work methods, and in this process they should remember their past experiences, and if possible have access to other members’ past experiences. Hence, the PAs must capture and represent the team members’ operations assisting the team members in the process of preserving and creating knowledge.

A PA works jointly with the organizer agent, a service agent capable of categorizing documents using the concepts from three ontologies: (i) a domain ontology, containing the main concepts of the domain of the project; (ii) a document ontology, holding the concepts concerning the documentation domain (e.g.
types of documents—a report, memo); and (iii) a project task ontology (e.g. definition of the requirements, prototype developing).

We have stated that documents are locally organized accordingly to the user preferences. The user preferences are expressed as a function of concepts coming from the ontologies. Team members can extend the ontologies (i.e. add children concepts). In this way, they can express their preferences, for instance, the concept “report” from the documentation ontology can be specialized by “task final report” or “task current status report”.

A service agent called project agent deals with all the various ontologies. The main goal of such an agent is to keep all the values shared by the group. The first action a personal assistant has to do when it integrates a project coterie is to ask the project agent for a copy of the ontologies.

The architecture is supposed to be reliable, secure and persistent. Thus, we integrated a service agent called repository agent that encapsulates a groupware tool or a database in order to benefit from the qualities (as they are proven technology) and from the services such tools provide (e.g. access control, persistence). Currently, we have established that the user can assign one of the following levels of sharing to his documents: private, project, and public. The repository agent must be able to translate such levels of confidentiality to the levels offered by the encapsulated tool. Besides access control, the repository agent offers services for saving and retrieving documentation, and, depending on the tool it encapsulates, may offer other kinds of services like Web search, or e-mail management.

Next, we describe how the PA and the service agents—the project, repository, and organizer agents—are arranged in a coterie.

4.2. The KM coterie

Although the architecture is not hierarchical, the service agents in our coterie have two scopes of action. Some of them are dedicated to one and only one user, and others, in the project scope, can serve all the agents belonging to the coterie.

The PA and the set of service agents dedicated to a single user form what we call the user’s staff (see Fig. 2). Such service agents cannot serve other users.

For example, user A’s organizer agent cannot organize the documents for user B, although it can answer queries from B. Agents belonging to the same staff usually run on the same computer.

In the project scope (the gray area in Fig. 2), one finds one transfer agent, one project agent, and one repository agent. Other SAs may be added or excluded at any time as needed. Service agents in the project scope usually run on different computers.

We have proposed four kinds of agents to be part of the KM system: personal assistant, organizer, repository and project. A fifth one, the transfer agent, may be present whenever we wish to connect a coterie with another coterie or with a different but FIPA compliant platform. Further details about the internal functioning of each kind of agent are given in the next section.

5. The KM in practice

This section describes a simple case giving further details on the internal functioning of the agents as well as on their interaction. We illustrate how the agents help the users to incrementally formalize their knowledge and to share it, and how agents can make team members aware of the other members’ activities. For the sake of clarity, we use the following example.

Fred, an employee of a certain enterprise that wants to develop KM, is part of a project called first approach to KM. We assume that he is currently working on the task prototype building and that the prototype will be developed using a groupware tool that is available to the team. Fred does not know the programming language provided by the groupware tool very well, and he is in need of detailed information on, let us say, refreshing automatic computed fields inside a form. Thus, he performs a number of operations. Firstly, he searches for a tutorial on the Web, finds an interesting one and saves it. Next, he chats with Sheila for getting detailed information on the problem, and finally he sends an e-mail to Peter asking for other sources of information. As he has spent some time in finding all this information, he believes it could be useful to keep the gathered information in a LL format. But, as he has no time to formalize this experience, all that information will be probably dispersed and lost in a few days. The following paragraphs explain how the KM system aids
Fred to preserve the experience and how other team members will be able to reuse it.

5.1. Capturing and representing desktop operations

While Fred is searching for information and dialo-
ging with his peers, the PA captures the operations he carries out on the computer desktop. Such operations are mainly related to communication (e.g. sending e-
-mails), and documentation (e.g. searching for docu-
ments) and they can be activated from the user inter-
face provided by the PA. The PA builds a
representation for the desktop operation as illustrated in Fig. 3. Note that there is always a document associated to an operation, for example, the TipsSheila.txt is a text file containing the dialog between Fred and Sheila.

Once the PA has partially represented the operation, it is transferred to the organizer agent that will complement the representation by identifying the relevant terms of the associated document. To compute the relevance of a term the organizer agent employs a simple measure based on the term frequency: the more often a term occurs in a document the more important it is. The organizer agent computes the measure only for terms corresponding to the concepts of the domain ontology.

5.2. Clustering operations

After some time, the Fred’s organizer agent has gathered a number of operations, together with the associated documents. We recall that the goal of such an agent is to organize Fred’s documentation, and that one of the goals of the system is to help Fred to articulate tacit knowledge. Hence, the organizer agent clusters operations attempting to correlate them with an implicit intention, i.e. the documents are correlated because they are relevant for reaching a goal that is not explicitly represented. In our case, Fred assigned himself a goal but he did not declare it to the system. May be, he has mentioned the goal to his colleagues in an informal way but this may be lost. Thus, clustering operations has a twofold purpose: to organize Fred’s documents by the similarity of their relevant terms, and to help Fred to make his intentions explicit.

The organizer agent keeps the operations in a frame structure, more precisely in memory organization packages [15]. For each new operation, it searches a cluster into which to fit the operation, using an incremental unsupervised learning algorithm called COBWEB [16]. Fig. 4 shows the clusters built by the organizer agent based on the relevant terms of the three operations Fred has performed: find a tutorial, chatting with Sheila, and sending an e-mail to Peter.

Note that although we know that the three operations of Fig. 4 should be clustered together, operation 1
is isolated because the relevant terms do not match the terms of the other two operations. In this case, the user may modify the proposed task moving the operation 1 to the desired group. Thus, even if the user does nothing, at least the system has organized the documents by indexing them automatically.

5.3. Transforming a cluster into tasks or LL

When Fred thinks that a cluster is well formed, he may decide to transform it into a task or an LL. The organizer agent may also notify Fred that it may be interesting to convert a certain cluster. The organizer agent learns to transform the clusters into tasks or LLs, and from time to time checks if a cluster can be transformed into a task/LL. The PA will aid Fred to reuse the information from the cluster in the construction of a task or a LL. The construction of either one is performed in a similar fashion.

Suppose that Fred grouped the three operations together and decided to transform the resulting cluster into an LL. Next, either Fred chooses the LL type that best fits the cluster or the organizer agent makes a suggestion based on past transformations. The LL and tasks types are defined in the R&D tasks ontology as well as their slots. Fred is asked to fill in the slots the system cannot obtain from the selected cluster automatically as the textual descriptions. In turn, the system is able to give the solution part of the LL filling it with the relevant resources (involved people and documents) referred to in the clustered operations as the arrows in Fig. 5 point out. After that, the system, using the domain ontology, performs a lexical analysis of the textual descriptions entered by Fred, and extracts terms that will index the LL. Such indexes in conjunction with the context box (project name, project task, involved people) will enable Fred and other team members to retrieve the LL information.

5.4. Exploiting and sharing the knowledge items

Over time, team members accumulate experiences and their organizer agents arrange them as LLs, tasks, and clusters. We refer to these elements as knowledge items or simply item when the context is clear. The sharing of such items occurs in the following situations: (i) a user makes a request to his PA; (ii) a PA makes a request to other PAs; or (iii) a PA pro-actively

```
Cluster
Operation 1
Type: SAVE-DOC
Name: save-doc-1
Owner: Fred
Terms: form, view, DB
Document: tutorial.html

Operation 2
Type: CHAT
Name: chat-1
Owner: Fred
Participants: Sheila
Terms: field, computed, form
Document: TipsSheila.txt

Operation 3
Type: SEND-EMAIL
Name: send-email-1
Owner: Fred
Participants: Peter
Terms: field, computed,
Document: toPeter.txt

Arrows indicate reuse of information in the LL

LL
Problem description: computed fields not refreshed when form is accessed on WEB.
Type: solving-a-problem
Context:
project name: first approach to KM project task: prototype building involved people: Fred, Sheila, Peter
Date: 4/3/02
Solution:
Textual:
Sheila has various programming References.
Resources:
Documents: tutorial.html, toPeter.txt, TipsSheila.txt
People: Sheila, Peter
```

Fig. 5. Reusing cluster information in a lesson learned.
suggests a past experience to the user. The first situation includes the second one, thus the latter will not be described.

5.4.1. Explicit user’s request to the PA

Some time past after defining the LL of Fig. 5, Fred wants to retrieve it. Thus, he sends a query to his PA that transfers the query to the organizer agent. The organizer agent traverses the past knowledge items evaluating their similarity with Fred’s query. In order to do that, we are currently implementing a textual case-based reasoning engine as proposed by Burke et al. [17] in the FAQ finder system.

When the organizer agent finds a knowledge item, it returns it to the PA. If the organizer agent has not found anything, it broadcasts a request to all organizer agents within the coterie. When all the agents have answered or the time limit for answering expires, the caller agent sorts the answers according to Fred’s profile and transmits them to the PA. Finally, the PA shows the answers to Fred.

The acquisition of LL allows to build a knowledge map containing the competences of the team members. The solution part of an LL contains information about people who contributed to solve the problem. Hence, we can associate them to the domain ontology concepts used for indexing the LL building, this way, the knowledge map.

5.4.2. Pro-active suggestion

Fred’s PA may set up the suggestion mechanism for suggesting knowledge items related to the current context of the work. The cluster that receives most part of the operations gives the current context. Thus, the organizer agent takes the relevant terms for the cluster, builds a query using the terms, and executes the process described for the first two situations. When the organizer agent has obtained the answers, it notifies the PA that can display the information to Fred.

5.5. Awareness

Suppose that while Fred is looking for information to solve a certain problem, and at same time two colleagues exchange a message concerning that problem. Fred’s PA can examine the message and make Fred aware that other colleagues work on the same subject or know something about the problem he is trying to solve. Of course, access control to the information must be addressed in this kind of monitoring. This mechanism is directly inspired from the concept of Ba [2].

Another way of making team members conscious of the others’ activities is by comparing their current contexts. As we briefly mentioned before, the cluster that receives most part of operations gives the current context. In fact, the context is composed of the relevant terms of that cluster, and by the current project task if available. PAs can broadcast a request for similar contexts and notify their respective users that a similar context has been found.

5.6. Validation

The proposed system is based on the following three points.

1. Integration: the KM system is supposed to be integrated into the daily activities of the project team members. The KM activity is not supposed to represent an overhead of work to the team members.
2. Types of knowledge: the proposed system is supposed to handle explicit knowledge as well as to assist team members to articulate tacit knowledge.
3. Distributed KM: the proposed system is supposed to handle different views of same knowledge items what incites evaluation of existing knowledge.

We would like to evaluate if the proposed system really contributes to the KM of a group when compared to a system that does not consider those three points. For measuring this, a controlled experiment will be performed. For example, two groups performing the same set of collaborative activities (activities belonging to the same domain), one group working with the proposed system, and the other group with ordinary tools for organizing documentation and communicating. We will compare both groups on the basis of the following factors:

- Time to accomplish the set of activities. It is interesting to observe if time decreases for the group using the proposed system, meaning that past experiences are really useful and reused in the subsequent activities.
- Number of created LL/tasks, and number of accesses. The goal is to determine the correlation
between the formalization of information and the time to execute the activity.

- Quality of the result. Although this is a subjective measure, it should nevertheless be measured to check if the workload between the two groups is balanced.

5.7. Important points

Several important points should be noticed.

Firstly, in the proposed system, information can be formalized gradually [1]. Thus, Fred can work with isolated operations and documents, tasks, or LLs. The clustering of operations aims at making the creation of LLs and tasks easier. The creation of LLs/tasks reduces the information overload and improves the accuracy of the returned answers, when later querying the system. Besides, in a more general perspective, it is an attempt to help team members to articulate part of their tacit knowledge and to stimulate team members to share their knowledge.

Secondly, Fred shares his experiences in a transparent way and is aware of what team members are doing. While Fred is working, the organizer agent is able to answer queries coming from other agent staffs. In the other direction, Fred’s staff is able to capture LLs from other members or send requests for information to them in order to make suggestions to Fred. Such a mechanism emulates what happens in a real team: each team member performs his own tasks and when he misses some information (or knowledge) to complete the task, he asks his colleagues for help.

Finally, our distributed approach to KM allows implementing a cooperative process of learning. When a knowledge item is reused, and hence put into a new context, the knowledge producer can evaluate the quality of the knowledge when other people reuse it. Thus, the producer can enhance and update the knowledge items. In addition, the producer can assist the customer in interpreting the knowledge item [3]. Future work will include the implementation of this idea.

6. Related work

Among the many issues we are currently addressing, we would like to focus on: (i) the capture of actions through PAs; (ii) the multi-agent systems that support KM systems; and (iii) the gradual formalization of the information.

PA applications may range from Internet search to collaborative tasks. In Lieberman et al. [18] and Huhns and Singh [19] a PA is used to search the Internet. While the user navigates, the agent proposes new Web sites analyzing the contents of the sites visited by the user. PAs are also used to improve the interface with complex systems. In Lieberman [20] a PA has been used to facilitate the interaction between the user and a Mac environment. The agent captures the events generated by the user while interacting with the operating system (for example, opening a directory, opening a text editor) and tries to foresee forthcoming events. In this manner, the system can suggest actions to the user saving her time. Maes (1994) reports successful application of agents in assisting user with email management, filtering news, scheduling meetings and selecting entertainment through social filtering. Our approach mainly differs1 from the cited ones because our PAs act cooperatively.

Other multi-agent systems have been developed for organizing group memories. The corporate memory management through agents (CoMMA) [21] project, combines emergent technologies (e.g. XML, machine learning) allowing users to exploit an organizational memory. The FRODO project [22] provides an agent-based middleware for distributed organizational memories whose main features are scalability and distribution. Work about digital libraries [23] where agents are responsible to keep track and give a uniform view of various documents stored in different databases, have some similarities mainly in relation to the document organization and retrieving. The main difference with our work is in the concept of coterie where PAs are capable of capturing circulating information whenever they believe it will be useful for their master.

In relation to the gradual formalization of information, the work of Shipman and McCall [1,5] has inspired us. The HOS system enables users working in a collaborative space to progressively formalize textual information in identifying objects of the domain and their attributes. An interesting point is

---

1 In Maes’s work on social filtering the agents act cooperatively.
that informal and formal information coexists, meaning that formalized information does not take the place of the corresponding informal one.

7. Conclusion

We analyze the proposed system from the KM perspective and from an architecture perspective for KM systems.

From the KM perspective, the proposed system:

- helps the user in articulating tacit knowledge and formalizing it gradually;
- reduces the information load because the more formalized the information, the more precise the system will be at retrieval time;
- augments knowledge sharing given that agents are capable of exchanging and capturing LL behind the scene; moreover, a staff of agents is able to proactively make suggestions to the user based on his current task; and
- allows team members to organize documentation according to their preferences by extending the document and domain ontologies.

The main drawback is that LL is just one of the ways of articulating tacit knowledge. Other methods should be considered, like the expression of the design rationale using, for example, decision trees.

Concerning the proposed KM system, we can say that the proposed architecture:

- defines a minimal MA architecture for KM systems composed by agents with two scopes of action, project and user;
- introduces the notion of an agent staff dedicated to a user in the same computer, which is somehow new; and
- encapsulates a groupware or database providing information access control to project documents (not to the messages exchanged among agents).

The prototype was first undertaken during another R&D project in the domain of automobile design, when we could test some ideas like textual CBR for retrieving segments of documents and a mechanism for suggesting similar documents according to what the user types into a document being edited. We are currently completing the multi-agent version of the prototype as described in this paper, more precisely implementing the clustering algorithm. We have to examine whether the use of extendible ontologies in per user mode causes problems of communication when the agents exchange messages. Another important issue is the cooperative evaluation of acquired knowledge. When other people reuse a knowledge item and modify it, it may be interesting to allow the knowledge producer to revise the item and keep track of the modifications.

Acknowledgements

This work was funded in part by CAPES and CEFET-PR, two Brazilian institutions, through a scholarship granted to Cesar Tacla and by a CNRS project from the PROSPER program.

References


Cesar A. Tacla is a research assistant at Centro Federal de Ensino Tecnológico do Paraná (CEFET-PR, Brazil) and currently he is a doctoral candidate in the Computer Science Department at the Université de Technologie de Compiègne (France). He received his BS in computer science from Universidade Federal do Paraná (Brazil) in 1990 and his MS from CEFET-PR in 1993. His research involves multi-agents systems in the support of knowledge management for research and development teams, specifically in automatic document organization and knowledge preservation.

Jean-Paul Barthès obtained his engineering degree from Ecole Centrale de Paris (France) and his PhD from Stanford University. Currently, he is a professor in the Department of Computer Science at the University of Technology of Compiègne (UTC) in France. He is the project director of the DICIT program, a distance learning program in the domain the design of technical documents, and of the e-COOP inter-laboratory research platform in the domain of cooperation and training. His main research interests are related to distributed artificial intelligence and mixed societies of cognitive artificial and human agents.