IMPLICT CULTURE-BASED PERSONAL AGENTS FOR KNOWLEDGE MANAGEMENT

Enrico Blanzieri, Paolo Giorgini, Fausto Giunchiglia
and Claudio Zanoni

May 2003

Technical Report # DIT-03-026

Implicit Culture-based Personal Agents for Knowledge Management

Enrico Blanzieri, Paolo Giorgini, Fausto Giunchiglia and Claudio Zanoni

Department of Information and Communication Technology
University of Trento - Italy
via Sommarive 14, 38050 Povo Trento
{enrico.blanzieri,paolo.giorgini,fausto.giunchiglia,claudio.zanoni}@dit.unitn.it

Abstract. We present an implementation of a multi-agent system that aims at solving the problem of tacit knowledge transfer by means of experiences sharing. In particular, we consider experiences of use of pieces of information. Each agent incorporates a system for implicit culture support (SICS) whose goal is to realize the acceptance of the suggested information. The SICS permits a transparent (implicit) sharing of the information about the use, e.g., requesting and accepting pieces of information.

1 Introduction

In Knowledge Management, knowledge is categorized as being either codified (explicit) or tacit (implicit). Knowledge is said being explicit when it is possible to describe and share it among people through documents and/or information bases. Knowledge is said being implicit when it is embodied in the capabilities and the abilities of the members of a group of people. Experience can be seen as a way for accessing and sharing this kind of knowledge. In [10], knowledge creation processes have been characterized in terms of tacit and explicit knowledge transformation processes, in which, instead of considering new knowledge as something that is added to the previous one, they conceive it as something that transforms it. Supporting by means of software systems the transfer of tacit knowledge among people in an organization represents a challenge whose difficulties are mainly in the need of explicitly representing tacit knowledge.

In [2], we have introduced the notion of Implicit Culture that can be informally defined (see Appendix A for a formal definition) as the relation existing between a set and a group of agents such that the elements of the set behave according to the culture of the group. Systems for Implicit Culture Support (SICS in the following) have the goal of establishing an Implicit Culture phenomenon that is defined as a pair composed by a set and a group in Implicit Culture relation. Supporting Implicit Culture is effective in solving the problem of improving the performances of agents acting in an environment where more-skilled agents are active, by means of an implicit transfer of knowledge between the group and the set of agents. In particular, Implicit Culture can be applied successfully in the context of knowledge management. The idea is to build systems able to capture implicit knowledge, but instead of sharing it among people, change the environment in order to make new people behave in accordance with this knowledge. As a first step in this direction we have showed how information retrieval problem can be posed in the implicit culture framework and how the framework generalizes collaborative filtering [4]. In this framework, supporting an Implicit Culture phenomenon leads to a solution of the problem of transfer tacit knowledge without the need to explicitly representing the knowledge itself.

Some assumptions underlie the concepts of Implicit Culture, Implicit Culture Phenomenon and SICS. We assume that the agents perform situated actions and perceive and act in an environment composed of objects and other agents. Before executing an action, an agent faces a scene formed by a part of the environment and it executes an action in that given situation. After a situated action has been executed, the agent faces a new scene. At a given time the new scene depends on the environment and on the situated executed actions. The action that an agent execute depends on the agent's state and, in general, it is not deterministically predictable with the information available externally. Rather, we assume that it can be characterized in terms of probability and expectations. Another assumption is that the expected situated actions of the agents can be described by a cultural constraint theory. Given a group of agents we suppose that there exists a theory about their expected situated actions. Such a theory can capture knowledge and skills of the agents about the environment and so it can be considered a cultural constraint of the group. Agents and objects, (i.e., the environment), are specified for each application.

The goal of a SICS is to establish an implicit culture phenomenon. The general architecture we have proposed in [2] (Figure 1) allows to establish an implicit culture phenomenon by following two basic steps: *(i)* defining a cultural constraint theory $\mathbf{\Sigma}$ for a group G and *(ii)* proposing to a group G' a set of scenes such that the expected situated actions of the set of agents G' satisfies $\mathbf{\Sigma}$. Both steps are realized by using the information about the situated executed actions of G and G'. An implementation of a SICS has been presented and showed to be effective in [3] and [4].

In this paper, we propose a multi-agent system for knowledge management where each agent incorporating the SICS contributes to propagate the information about the actions of the user to other agents. As presented in [2], the SICS can be seen as a generalization of a memory-based collaborative filtering. The system adopts a distributed approach to knowledge management opposed to a centralized one as pointed out in [5].

The paper is organized as follows. Section 2 and Section 3 present the multi-agent architecture and the implementation of the SICS, respectively. Section 4 describes related work and Section 4 draws conclusions a future directions. Finally, in order to facilitate the reading, Appendix A recalls the formal definition of Implicit Culture presented in [3].

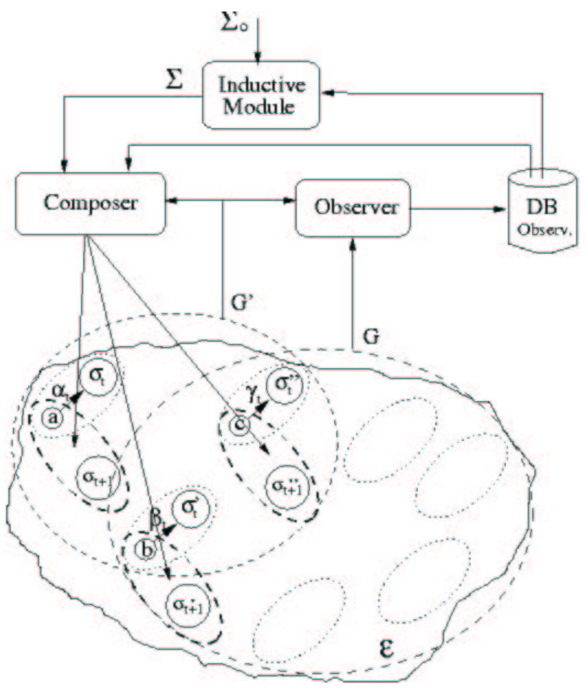

Fig. 1. The basic architecture for Systems for Implicit Culture Support consists of the following three basic components: *observer* that stores in a data base (DB) the situated executed actions of the agents of G and G' in order to make them available for the other components;*inductive module* that, using the situated executed actions of G in DB and the domain theory $\mathbf{\Sigma}_0$, induces a cultural constraint theory $\mathbf{\Sigma}$; *composer* that, using the cultural constraint theory $\mathbf{\Sigma}$ and the executed situated action of G and G', manipulates the scenes faced by the agents of G' in such way that their expected situated actions are in fact cultural actions with respect to G. As a result, the agents of G' execute (on average) cultural actions w.r.t. G, and thus the SICS produces an Implicit Culture phenomenon.

2 A Multi-agent System based on Implicit Culture

In this section we present the multi-agent system based on the Implicit Culture we have developed for Knowledge Management applications. The system has been built using JADE (Java Agent Development Framework) [1], a software development framework for developing multi-agent systems conforming to the FIPA standards [8]. Basically, the system is a collection of personal agents that interact one another in order to satisfies the requests of their users. Each agent uses locally the SICS to suggest both its user and the other agents. Applying the SICS locally, each personal agent is able to provide suggestions from its perspective, namely on the base of the information it has collected observing the behavior of its user and those of the agents with which it has interacted with. In our system we have extended the FIPA protocols in order to allows the agents to

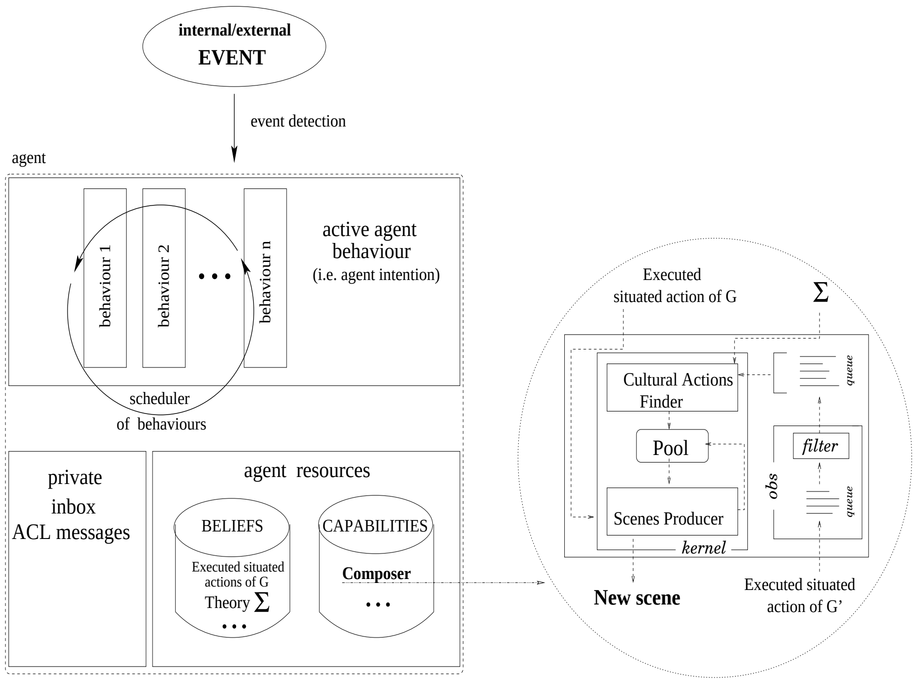

Fig. 2. Internal architecture of a JADE agent implementing a SICS

exchange each other feedback about how the users use the information suggested by their personal agents.

Using the system, a user asks her personal agent about a keyword and the agent starts to search for documents, links, and references to other users, related to the keyword. The personal agent tries to suggest the user using the SICS and the observations done in the past on the user's behavior and on the behavior of the users whose personal agents it interacted with. Alternatively, the personal agent can submit the request to other agents which will treat the request as it were done by their users. In this case, however, the suggestions can include also other agents to contact. The selection of the agents to send the request is done applying locally the SICS again.

Figure 2 presents the general architecture of each single personal agent implemented with JADE. The architecture of a JADE agent consists of four main components: *Behaviors*, *Scheduler*, *Inbox* , and *Resources*. In our implementation we have:

– *Behaviors*, that an agent is able to adopt in response to different internal and external events. All tasks are implemented as behavior objects; we have a specific behavior for the SICS. A request from the user or from another agent actives the SICS behavior.

– *Scheduler*, that determines which behavior is the current focus of the agent and consequently it selects an action to perform.
– *Inbox* , a queue of incoming messages (ACL). It contains the messages coming from the user as well as those from other agents.
– *Resources*, consisting of beliefs and capabilities. The agent's beliefs are the information available to the agent and the capabilities are particular functionalities used in the behaviors. In our implementation the three main components of the SICS (observer, composer and inductive module) are three different capabilities and the observations and the cultural constraint theory are stored as beliefs. Additionally, each personal agent has beliefs about a local schema useful to organize the information available. This schema is not mandatory.

```
<?xml version="1.0"?>
<tree name="USER">
   <node name="travels">
      <node name="train timetable">
         <node>
            <name>www.fs-on-line.it< /name>
            <type>http< /type>
         < /node>
         <node>
            <name>info@trenitalia.it< /name>
            <type>mailto< /type>
         < /node>
      < /node>
   < /node>
< /tree>
```

Fig. 3. An example of local schema expressed in XML

The capability (the composer) and the beliefs (situated executed actions and cultural constraint theory) related to the SICS and reported in Figure 2, will be presented in details in the next section. Here we concentrate on the other beliefs and behaviors. Each personal agent has among its beliefs a local schema in order to organize information available to its user. Basically, the schema is a tree where the nodes are labeled with strings that the user uses to describe her own areas of interest, and the leaves are links. A link can be a reference to a document stored locally in the user system or it can be an Internet address or a reference to a person (e.g., a phone number, an email address or just the name of the person). The schema is a conceptual representation of how the user organize locally its information, and it does not say anything about how this representation matches with those of the other users. The schema is represented in XML (see Figure 3 for an example).

Figure 4 shows the algorithm used by personal agent when it receives a request of information from its user or from some other agent. The global variable `result` contains both links and names of the agents of the platform. If the message is a query the SICS behavior is activated and it modifies `result`; if no agents appear in `result` the DF agent is added to it in order to propagate the query in any case; if the sender of the query is the user the link contained in `result` are sent back and a query is sent to all the agents contained in `result`. If the message is a reply from an agent the complete `result` (links and agents) is sent, whereas an incomplete `result` (links only) is sent in the case the reply comes from the user.

```
global result
for all message in INBOX do
  if (message.type == 'query') then
     result := nil
     SICS-behavior(query.sender,query.content,
     result.links,result.agents)
     if (result.agents == nil) then
       add(DF,result.agents)
     end if
     if (query.sender == user) then
       inform(self,user,result.links)
       for all result.agent do
         request(self,result.agent,query.content)
       end for
     end if
  else if (message.type == 'reply') then
          if (reply.sender == user) then
             inform(self,user,result.links)
          else inform(self,message.sender,result)
          end if
       end if
  end if
end for
```

Fig. 4. The algorithm used by the personal agent for processing the messages

The agents interact one another using the FIPA-Iterated-Contract-Net Protocol, that starts with a call for proposal to perform a given action. In particular, we use the call for proposal for checking the availability of an agent to perform a search action. Differently, the user interacts with its personal agent using the the FIPA-Query Protocol. Additionally, we have introduced a third protocol for the propagation of the user feedback about the suggestions provided to him.

In particular, the protocol guarantees that the user informs the personal agent about the acceptance of the refusing of a suggestion, and that the personal agent informs about this the other agents it asked. In practice, the sending of an inform whose content is "accept" is triggered by an action of the user, e.g., following a link, maintaining it implicit.

An example of interaction. Let consider the case in which a *user* searches information about *"train timetable"* and asks his *personal agent*. Let suppose that the SICS suggests an Internet address (*www.fs-on-line.it*) and another agent, *agent-1*. The personal agents informs the user about the address *www.fs-on-line.it* and send a request to *agent-1*. Supposing that *agent-1* replies with another internet address *www.trenitalia.it* and another agent, *agent-2*, then the *personal agent* will send a request to *agent-2*. When *agent-2* replies with th email address *info@trenitalia.it*, the *personal agent* informs the user with the results it has collected (namely, *"www.fs-on-line.it"* + *"www.trenitalia.it"*+ *"info@trenitalia.it"*). Finally, if the *user* executes an action considered of acceptance for example of *"info@trenitalia.com"* an inform with that content is sent. The *personal-agent* informs *agent-2* because it has suggested such an address, and *agent-1* because it has suggested *agent-2*. Figure 5 presents the sequence of messages exchanged by the agents.

```
 1. request(user,personal-agent,``train timetable'')
 2. inform(personal-agent,user,``www.fs-on-line.it'')
 3. request(personal-agent,agent-1,``train timetable'')
 4. inform(agent-1,personal-agent,``www.trenitalia.it''+``agent-2'')
 5. request(personal-agent,agent-2,``train timetable'')
 6. inform(agent-2,personal-agent,``info@trenitalia.it'')
 7. inform(personal-agent,user,``www.trenitalia.it''+``info@trenitalia.it'')
 8. inform(user,personal-agent,``accept(info@trenitalia.it)'')
 9. inform(personal-agent,agent-1``accept(info@trenitalia.it)'')
10. inform(personal-agent,agent-2,``accept(info@trenitalia.it)'')
```

Fig. 5. An interaction example

The example shows how the variant of the FIPA communication protocol permits to the agents to propagate the feedback of the user. In this way each personal agent has access locally to information about the use of the information done by the requester. The availability of the information permits to the agent to observe a wider number of actions permitting the transfer of knowledge between the users. Indeed, if the personal agent would limit its observations only to the actions performed by its user, the effect achieved by the user would be a simple personalization. With the communication protocol we have adopted, each SICS can observe also actions done by the users of the personal agents it has been put in contact to. It is worth to note that this is transparent to the user. As a summary, the personal agent act on behalf of the user in a complex way. It uses the observations of the behavior of its user to provide a better service to the user herself (personalization) and to the other users (collaboration). Moreover, with the same goal, it integrates locally the observations of the user with the observations of the other users and contribute to propagate the observations of its own user in order to give feedback to the other agents. In other terms the user delegates to the personal agent the capacity of sharing information about the use of information.

3 The implementation of the SICS behaviors and capability

The SICS we have implemented and inserted in the agents as behavior and capability of JADE, is a particular case of the general one. Observations are treated as beliefs that are updated depending on the type of messages. Moreover, we do not consider any kind of theory induction over the observations, the cultural constraint theory is completely specified and the inductive module is omitted (i.e., in Figure 1, $\mathbf{\Sigma} \equiv \mathbf{\Sigma}_0$). The cultural constraint theory is expressed by a set of rules of the form:

$$A_1 \wedge \cdots \wedge A_n \rightarrow C_1 \wedge \cdots \wedge C_m$$

in which $A_1 \wedge \cdots \wedge A_n$ is referred to as the antecedent and $C_1 \wedge \cdots \wedge C_m$ as the consequent. The idea is to express that "if in the past the antecedent has happened, then there exists in the future some scenes in which the consequent will happen". Hence the consequents has to be interpreted as situated expectations. Antecedent and consequent are conjunctions of atoms, namely two types of predicates: observations on an agent and conditions on times. For instance, $\texttt{request}(x, y, k, t_1)$ is a predicate of the first type that says that the agent x requests to agent y informatin relevanto to the keyword k at the time t_1; while $\texttt{less}(t_1, t_2)$ is an example of the second type and it simply states that $t_1 < t_2$.

In our application the cultural constraint theory is fixed a priori and very simple. Indeed, we want each personal agent PA to recommend links or agents that satisfy the request, namely that the expected situated action of the user (and consequently of her personal agents in the system) is to accept the recommendation of the agent PA. The following rule is used to express the cultural theory:

$$\begin{array}{l}\texttt{request}(x, \texttt{PA}, k, t_1) \wedge \texttt{inform}(\texttt{PA}, x, y, t_2) \wedge \texttt{less}(t_1, t_2) \rightarrow \\ \texttt{accept}(x, y, k, t_3) \wedge \texttt{less}(t_2, t_3)\end{array} \quad (1)$$

which states that if x (user or agent) asks the PA information relevant to the keyword k, and the PA replies informing x that y (link or agent) are relevant, then x will accept from y information as relevant to the keyword k. In other terms, the theory specifies that the agents should accept the information they are offered. Each agent has the goal of having the group of agents and users behaving consistently with the theory. This goal is achieved by using the composer of the SICS architecture.

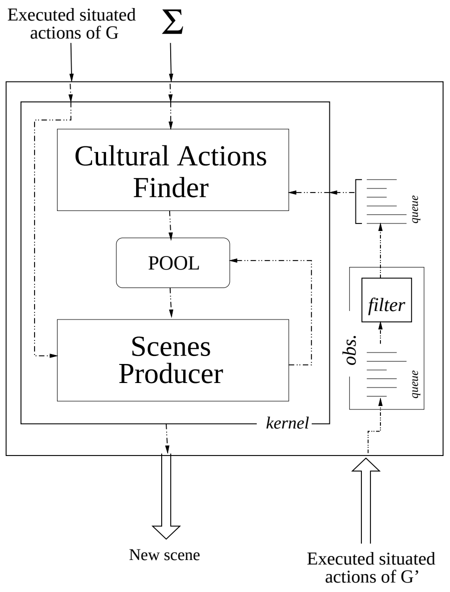

Fig. 6. The composer architecture

The goal of the composer is to propose a set of scenes to agents of G' such that the expected situated actions of these agents satisfy the cultural constraint theory $\mathbf{\Sigma}$ for the group G. In our implementation, the composer consists of two main submodules (Figure 6)[1]:

- the *Cultural Actions Finder* (CAF), that takes as inputs the theory $\mathbf{\Sigma}$ and the executed situated actions of G', and produces as output the cultural actions w.r.t. G (namely, the actions that satisfy $\mathbf{\Sigma}$).
- the *Scenes Producer* (SP), that takes one of the cultural actions produced by the CAF and, using the executed situated actions of G, produces scenes such the expected situated action is the cultural action.

Cultural Actions Finder

The CAF matches the executed situated actions of G' with the antecedents of the rules of $\mathbf{\Sigma}$. If it finds an action that satisfies the antecedent of a rule, then it takes the consequent of the rule as a cultural action. Figure 3 presents the algorithm for the CAF. For each rule `r` (`ant`$\rightarrow$`cons`), the function *match*(`a`,α) verifies whether the atom `a` of `ant`=*ant*(`r`) matches with the executed situated action α; then the function *find-set*(`ant`,`past-actions`) finds a set `past-actions` of past executed situated actions that matches with the set of atoms of `ant`; and finally,

[1] An additional component of the composer is the *Pool*, which manages the cultural actions given as input from the satisfaction submodule. It stores, updates, and retrieves the cultural actions, and solves possible conflicts among them.

```
loop
  get the last executed situated action $\alpha$
  for all rule r of $\Sigma$ do
    for all atom a of ant(r) do
      if match(a,$\alpha$) then
        if find-set(ant,past-actions) then
          r'=join(past-actions,r)
          return cons(r')
        end if
      end if
    end for
  end for
  return false
end loop
```

Fig. 7. The algorithm for the CAF submodule

the function *join*(`past-actions`,`r`) joins the variables of `r` with the situated executed actions in `past-actions`. The function *cons*(`r'`) returns the consequent of `r'`.

Scenes Producer

Given a cultural action α for the agent $x \in G'$ that performed actions on the set of scenes $S(x)$, the algorithm used in the scenes producer consists of three steps:

1. find a set of agents $Q \subseteq G \cup G'$ that performed actions similar to α and the sets of scenes $S(y)$ with $y \in Q$ and in which they performed actions;
2. select a set of agents $Q' \subseteq Q$ similar to x;
3. Estimate (using Q') the expected similarity between the expected actions of x in the scenes of the set $S = \bigcup_{y \in Q} S(y)$ and the cultural action α. Return the scene that maximizes the expected similarity and propose it to x.

Figure 3 shows the simple algorithm used in step 1. An agent y is added to the set Q if the similarity $sim(\beta_y, \alpha)$ between at least one of its situated executed actions β_y and α is greater than the minimum similarity threshold T_{min}. The scenes s in which the β_y actions have been executed are added to $S(y)$, that is the set of scenes in which y has performed actions similar to α.

Step 2 selects in Q the k nearest neighbors to x with respect to the agent similarity defined as follows:

$$w_{x,y} = \frac{1}{|S(x) \cap S(y)|} \sum_{\sigma \in S(x) \cap S(y)} \frac{1}{|A_x(\sigma)||A_y(\sigma)|} \sum_{\beta_x \in A_x(\sigma)} \sum_{\beta_y \in A_y(\sigma)} sim(\beta_x, \beta_y) \qquad (2)$$

```
for all $y \in G'$
    for all situated executed actions $\beta_y$ of $y$
        if sim($\beta_y, \alpha$)> $T_{min}$ then {
            if $y \notin Q$ then  $y \rightarrow Q$
            $s \rightarrow S(y)$
        }
```

Fig. 8. The algorithm for step 1

where $S(x) \cap S(y)$ is the set of scenes in which both x and y have executed at least an action. $A_x(\sigma)$ and $A_y(\sigma)$ are the set of actions that x and y have respectively performed in the scene σ. Eq. 2 could be replaced by a domain-dependent agent similarity function if needed.

Step 3 selects the scenes in which the cultural action is the expected situated action. To do this, firstly we estimate for any scene $\sigma \in S = \bigcup_{y \in Q} S(y)$ the similarity value between the expected action of x and the cultural action, and then we select the scene with the maximum value. The function to be maximized is the expected value $E(sim(\beta_x, \alpha)|\sigma)$, where β_x is the action performed by the agent x, α is the cultural action, and $\sigma \in S$ is the scene in which β_x is situated. The following estimate is used:

$$\hat{E}\left(sim(\beta_x, \alpha)|\sigma\right) = \frac{\sum_{u \in Q'} E\left(sim(\beta_u, \alpha)|\sigma\right) * w_{x,u}}{\sum_{u \in Q'} w_{x,u}} \tag{3}$$

that is we calculate the weighted average of the similarity of the expected actions for the neighbor of the scene, the weight $w_{x,u}$ is the similarity between the agent x and the agent u, whereas $E\left(sim(\beta_u, \alpha)|\sigma\right)$ with $u \in Q'$ in Eq. 3 is estimate as follows:

$$\hat{E}\left(sim(\beta_u, \alpha)|\sigma\right) = \frac{1}{|A_u(\sigma)|} \sum_{\beta_u \in A_u(\sigma)} sim(\beta_u, \alpha) \tag{4}$$

that is the average of $sim(\beta_u, \alpha)$ over the set of actions $A_u(\sigma)$ performed by u in σ.

The algorithms described above, as well as the multi-agent system presented in the previous section, is fully implemented in Java using XML for expressing the cultural constraint theory.

4 Related Work

Different areas of research produced related work. The main areas we consider here are Agent-Mediated Knowledge Management (AMKM), Distributed Knowledge Management (DKM) and Computer Supported Collaborative Work (CSCW).

In AMKM area, Yu and Singh [12] have recently proposed an agent-based referral system. Their system, called MARS, suggests to the user the experts she might contact in order to satisfy her knowledge needs. Each user has a personal agent that interacting by means of a mail server with other personal agents supports the user's social network. As in our system, in MARS agents are provided with learning capabilities. However, our system differs from MARS in the system architecture as well as from the theoretical point of view. Architecturally, MARS agents learn explicit models of their *neighbors* and *acquaintances*, whereas our approach is memory-based and there is no explicit classification of the other agents; our system adopts FIPA standards and JADE platform and it is web-based and not mail-based. Theoretically, the presence of a shared ontology between the agents make MARS only partially distributed, because the ontology has to be fixed for all the agents. Moreover, we emphasize the implicit support of knowledge by managing documents, links and reference to people in a uniform way by inserting the implicit transfer of knowledge among the goals of the personal agents.

A purely distributed approach to knowledge management is being consistently addressed in the EDAMOK project [6, 7]. The system-development part of the project adopts a peer-to-peer architecture with an explicit notion of *context*. Based on the published material it is possible to sketch some differences. Architecturally, our agent-based approach relies on different architecture and technology and insert a learning functionalities in order to discover and propagate information about the other entities (agents or peers) in the system. Theoretically, their approach tends to solve *a posteriori* the problem of matching between the local perspectives (contexts) whereas our system tends to support the formation of compatible local perspectives.

In the area of CSCW, the management of tacit knowledge is receiving increasing attention [9]. Tacit knowledge is hard to be transmitted and shared in computer supported environment. Ribak et al [11] wrote:"By replacing face-to-face communication with telephone, e-mail, and instant messaging, we have also forfeited overhearing hallway conversations and the constant subconscious awareness of the state of our team and works environments". They ReachOut system allows for easy and partial-persistent question/answering conversational exchanges between groups of human peers. The effect is community building and awareness of the others. Tacit knowledge can be shared during the exchanges as it happens in face-to-face interactions. The main difference of our approach is that we aim to share tacit knowledge implicitly, namely without the need to communicate and be aware of the other users. The two approaches are complementary and could both gain from an integration.

5 Conclusions and future work

We have presented a multi-agent system that exploits the architecture of the Systems for Implicit Culture Support in order to solve the problem of the tacit knowledge transfer in a knowledge management context. We have argued that the tacit knowledge transfer requires the sharing of experiences and that the main difficulty relies in the need of explicitly representing the tacit knowledge. Our approach aims to by-pass the problem of the explicit representation.

The system incorporates a SICS in each agent. The SICS is used in order to provide information to the user and also to the other users by means of a communication protocol between the agents. The SICS observes the local actions of its own user and, by means of a variant of the FIPA communication protocols, also the actions of the other users. The multi-agent architecture permits the exchange of information about the users actions, improving so the range of the actions that each local SICS can observe. The overall effect is an implicit transfer of information about the use of the suggested items. In other terms, the system supports the sharing of the experience of the use of some pieces of information.

The proposed system represents a viable way of supporting the transfer of tacit knowledge between individuals in an organization. Each personal agent contributes locally to a realization of an implicit culture phenomenon. It is important to note that the local perspective of each agent permits the existence of different practices, given the fact that not all the agents will converge to the same set of observations and consequently to the same suggestions.

The agents' capabilities implemented in our system can be used in different scenarios. For example, instead of information searching, it is possible to apply our approach to a case-based reasoning scenario, where the goal is to find similar solutions for similar problems reusing solved cases. Each agent can suggest to the other agents the most proper case for a given problem. Some agents can wrap a case-base and implement different similarity functions that can be available in the platform by means of the Directory Facilitator.

The advantages of our approach include:

- each agent helps the user to deal with implicit knowledge, learn and interact with the other users;
- the system maintains knowledge implicit, namely implicit knowledge and behaviors are captured and explicitly represented by the system without need to explicit them to the user;
- the system supports the user in the learning activity. Each personal agent has learning capabilities with respect to the user's preferences;
- the system supports interaction and collaboration. Each personal agent exchanges information about how information have been used;
- the system is conceptually distributed. The distribution permits the emergence of local perspectives and learning capabilities permit to harmonize them without need of explicit capabilties for solving conflicts.
- the system can help to defend the investment on knowledge. An agent survives also in the case the user left the organization and it can be used as long as it proves to be competent.

Provided that our system is not alternative to other systems but it can be seen as an additional web-based service, it has few disadvantages:

- the users need to be willing to share information about how they use shared or public documents (note that we do not require documents sharing);

– the system does not observe all the possible search behaviors of the user, but only the ones performed by means of the system (namely, preserving privacy);
– technically, it requires a rather demanding message interactions between the agents. This can cause an overload in case of an high number of agents.

We are currently applying our system to a real situation, in particular we are working to a version of our system to support knowledge management in a business intelligence company.

References

1. F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with jade. In *Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000)*, Boston, MA, 2000.
2. E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture. In *Proceedings of the Workshop on Agents and Recommender Systems*, Barcellona, 2000.
3. Enrico Blanzieri, Paolo Giorgini, Paolo Massa, and Sabrina Recla. Implicit Culture for Multi-agent Interaction Support. In Carlo Batini, Fausto Giunchiglia, Paolo Giorgini, and Massimo Mecella, editors, *Cooperative Information Systems, 9th International Conference - CoopIS 2001*, volume 2172 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, 2001.
4. Enrico Blanzieri, Paolo Giorgini, Paolo Massa, and Sabrina Recla. Information Access in Implicit Culture Framework. In *Proceedings of the Tenth ACM International Conference on Information and Knowledge Management (CIKM 2001)*, Atlanta, Georgia, November 2001.
5. M. Bonifacio, P. Bouquet, and A. Manzardo. A Distributed Intelligence Paradigm for Knowledge Management. In *AAAI'2000 Spring Symposium on Bringing Knowledge to Business Processes*, Stanford University, Palo Alto (California, USA), 20-22 Marzo 2000.
6. Matteo Bonifacio, Paolo Bouquet, and Paolo Traverso. Enabling Distributed Knowledge Management. Managerial and Technological Implications. *Informatik/Informatique*, III(1), 2002. Special Issue on Knowledge Management, S. Lueg (ed.).
7. EDAMOK.
8. FIPA. Foundation for intelligent physical agents. http://www.fipa.org.
9. Michal Jacovi, Amnon Ribak, and Andree Woodcock. Managing tacit knowledge: a report on the european cscw workshop, 2001.
10. I. Nonaka and H. Takeuchi. *The knowledge Creating Company.* Oxford University Press, New York, 1995.
11. Amnon Ribak, Michal Jacovi, and Vladimir Soroka. "Ask before you search" Peer Support and Community building with Reachout. In *CSCW2002*, 2002.
12. Bin Yu and Munindar P. Singh. An Agent-Based Approach to Knowledge Management. In *Proceedings of Eleventh International Conference on Information and Knowledge Management (CIKM02)*, SAIC Headquarters, McLean, Virginia, USA, November 2002.

APPENDIX A: Formal Definition of Implicit Culture

We consider *agents* and *objects* as primitive concepts to which we refer with strings of type *agent_name* and *object_name*, respectively. We define the *set of agents* $\mathcal{P}$ as a set of *agent_name* strings, the *set of objects* $\mathcal{O}$ as a set of *object_name* strings and the *environment* $\mathcal{E}$ as a subset of the union of the set of agents and the set of objects, i.e., $\mathcal{E} \subseteq \mathcal{P} \cup \mathcal{O}$.

Let *action_name* be a type of strings, E be a subset of the environment ($E \subseteq \mathcal{E}$) and s an *action_name*.

Definition 1 (action). *An action α is the pair $\langle s, E\rangle$, where E is the argument of α ($E = arg(\alpha)$).*

Let $\mathcal{A}$ be a set of actions, $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{E}$.

Definition 2 (scene). *A scene σ is the pair $\langle B, A\rangle$ where, for any $\alpha \in A$, $arg(\alpha) \subseteq B$; α is said to be* possible in σ. *The* scene space $\mathcal{S}_{\mathcal{E},\mathcal{A}}$ *is the set of all scenes.*

Let T be a numerable and totally ordered set with the minimum t_0; $t \in T$ is said to be a *discrete time*. Let $a \in \mathcal{P}$, α an action and σ a scene.

Definition 3 (situation). *A situation at the discrete time t is the triple $\langle a, \sigma, t\rangle$. We say that a* faces *the scene σ at time t.*

Definition 4 (execution). *An* execution *at time t is a triple $\langle a, \alpha, t\rangle$. We say that a* performs *α at time t.*

Definition 5 (situated executed action). *An action α is a* situated executed action *if there exists a situation $\langle a, \sigma, t\rangle$, where a performs α at the time t and α is possible in σ. We say that a performs α in the scene σ at the time t.*

When an agent performs an action in a scene, the environment reacts proposing a new scene to the agent. The relationship between the situated executed action and new scene depends on the characteristics of the environment, and in particular on the laws that describe its dynamics. We suppose that it is possible to describe such relationship by an environment-dependent function defined as follows:

$$F_{\mathcal{E}} : A \times \mathcal{S}_{\mathcal{E},\mathcal{A}} \times T \rightarrow \mathcal{S}_{\mathcal{E},\mathcal{A}} \tag{5}$$

Given a situated executed action α_t performed by an agent a in the scene σ_t at the time t, $F_{\mathcal{E}}$ determines the new scene σ_{t+1} ($= F_{\mathcal{E}}(\alpha_t, \sigma_t, t)$) that will be faced at the time $t + 1$ by the agent a.

While $F_{\mathcal{E}}$ is supposed to be a deterministic function, the action that an agent a performs at time t is a random variable $h_{a,t}$ that assumes values in $\mathcal{A}$.

Let $a \in \mathcal{P}$ and $\langle a, \sigma, t\rangle$ be a situation.

Definition 6 (expected action). *The* expected action *of the agent a is the expected value of the variable $h_{a,t}$, that is $E(h_{a,t})$.*

Definition 7 (expected situated action). *The* expected situated action *of the agent a is the expected value of the variable $h_{a,t}$ conditioned by the situation $\langle a, \sigma, t\rangle$, that is $E(h_{a,t}|\langle a, \sigma, t\rangle)$.*

Definition 8 (party). *A set of agents $G \subseteq \mathcal{P}$ is said to be a* party.

Let $\mathcal{L}$ be a language used to describe the environment (agents and objects), actions, scenes, situations, situated executed actions and expected situated actions, and G be a party.

Definition 9 (cultural constraint theory). *The* Cultural Constraint Theory for G *is a theory expressed in the language $\mathcal{L}$ that predicates on the expected situated actions of the members of G.*

Definition 10 (group). *A party G is a* group *if exists a cultural constraint theory $\mathbf{\Sigma}$ for G.*

Definition 11 (cultural action). *Given a group G, an action α is a* Cultural Action w.r.t. *G if there exists an agent $b \in G$ and a situation $\langle b, \sigma, t\rangle$ such that*

$$\{E(h_{b,t}|\langle b, \sigma, t\rangle) = \alpha\}, \mathbf{\Sigma} \nvdash \perp$$

where $\mathbf{\Sigma}$ is a cultural constraint theory for G.

Definition 12 (implicit culture). Implicit Culture *is a relation $\rtimes\!\!\!\ltimes$ between two parties G and G' such that G and G' are in relation ($G \rtimes\!\!\!\ltimes G'$) iff G is a group and the expected situated actions of G' are cultural actions w.r.t G.*

Definition 13 (implicit culture phenomenon). Implicit Culture Phenomenon *is a pair of parties G' and G related by the Implicit Culture.*

We justify the "implicit" term of implicit culture by the fact that its definition makes no reference to the internal states of the agents. In particular, there is no reference to beliefs, desires or intentions and in general to epistemic states or to any knowledge about the cultural constraint theory itself or even to the composition of the two groups. In the general case, the agents do not perform any actions explicitly in order to produce the phenomenon.