Implicit Culture for Multi-agent Interaction Support

E. Blanzieri[1,3], P. Giorgini[2], P. Massa[1], and S. Recla[1]

[1] ITC-irst Trento - Italy - {blanzier,massa,recla}@itc.it
[2] Dep. of Mathematics - University of Trento - Italy - pgiorgini@science.unitn.it
[3] Present: Dep. of Psychology - University of Turin - Italy - blanzier@psych.unito.it

Abstract. Implicit Culture is the relation between a set and a group of agents such that the elements of the set behave according to the culture of the group. Earlier work claimed that supporting Implicit Culture phenomena can be useful in both artificial and human agents. In this paper, we recall the concept of Implicit Culture, present an implementation of a System for Implicit Culture Support (SICS) for multi-agent systems, and show how to use it for supporting agent interaction. We also present the application of the SICS to the eCulture Brokering System, a multi-agent system designed to mediate access to cultural information.

1 Introduction

Communication and autonomy are basic capabilities of artificial agents [12], and the agent paradigm suggests that the goals of a Multi-Agent System (MAS) should be reached via the interaction of autonomous communicating agents [11]. In general, interaction is a critical issue. In a brokering system [6, 10] for instance the goal of the broker is to facilitate the interaction, which intermediates access to the services of different agents. The overall efficiency of the system is improved through the existance of the broker, because agents can ask the broker for a service without needing to know who are the most suitable agents for performing the given task. In this case, a good interaction policy improves the performances of the system.

Let us consider two sets of agents acting in the same environment with the same goals, but with different skills for performing tasks. For example, in the brokering system we can consider a set of agents that have the capacity to refuse the services provided by unreliable or unsafe agents and another set of agents (or even a single agent) that do not have enough knowledge for doing so. Obviously the latter set of agents performs poorly due to its engagement in unproductive communication. This set of agents can decrease the interaction quality of the MAS even though the other group of agents has good knowledge and skills, i.e. a successful "culture".

The aim of this work is to explore ways of supporting interaction within MAS that takes advantage of the co–presence of sets of agents with different skills. In particular, we focus on the relation between two sets of agents, deriving a strategy and implementing an architecture for improving the interaction. In order to investigate the relation between sets of agents acting consistently, two of the authors introduced the concept of Implicit Culture [2] in a previous work. The definition of Implicit Culture captures the relation between two sets of agents such that the former behaves according to the culture of the latter. Supporting Implicit Culture is an effective way of improving the performances of agents acting in an environment where "better-skilled" agents are also active. Advantages of Implicit Culture support have also been proposed for artificial agents [3], in particular, for controlling the requirements of agent-based systems. Both [2] and [3] have introduced the general architecture of a System for Implicit Culture Support (SICS). No general implemented SICS was given however. In this paper, we present a SICS for multi-agent systems and we show how it is applied to a particular MAS, the eCulture Brokering System.

The paper is organized as follows. In section 2 we describe the concept of Implicit Culture. In section 3 we present a SICS for multi-agent systems, and we show how to use it for supporting agent interactions. Section 4 presents the eCulture Brokering System, a multi-agent system in which the SICS is used. Finally, section 5 gives some conclusions and describes future work.

2 Implicit Culture

When an agent begins to act in an environment, for which it has insufficient knowledge or skills, its behavior is far from optimal. The problems that the new agent must face are made even more complex if other agents are active in the same environment. Other agents would probably have more knowledge and be more skilled. Moreover, they might not be willing to share their knowledge or might not even be able to represent and communicate it. In order to improve its behavior, the new agent should act consistently with the culture of the other agents. Instead, this "new kid in town" is unable to cope with the new environment and with the other agents. Depressingly, a group of agents do have the knowledge and actively exploit it. In the case of humans the phenomenon is sometimes referred to as "cultural shock". Indeed, knowledge about the environment and about the behavior of the agents is part of their culture and that is what the new agent lacks.

The problem of getting the new agent to act consistently with the knowledge and the behaviors of the group could be solved by improving the capabilities of the agent in terms of communication, knowledge and learning. The first solution is just "to ask someone", but in a multi-agent setting, this is *not* a simple solution. In fact, it is necessary to know what to ask (knowledge about the problem), how to ask (a language for expressing the problem), and who to ask (some brokering facility). The second possible solution is to represent the relevant knowledge and to provide it directly to the agent. If the knowledge required is objective and relatively static, this representation can be made by observing the environment and describing it. Building ontologies is a common way to address this problem. Unfortunately, the environment can be partially unknown and intrinsically dynamic. As a third option, it is possible to equip the agent with both observational and learning capabilities allowing it to acquire skills by imitating other agents. The drawback of this techinque is that these capabilities are rather complex and their application requires resources.

When the environment is partially under control, the problem can be tackled in a very different way. Instead of working on the agents capabilities, it is also possible to modify the view that the agent has of the environment and thereby modify its actions. In fact, changing in a proper way the set of possible actions that the agent can perform in the environment can lead the agent to act consistently with the behavior of the group. The group itself can have its behavior optimized on the particular environment. Moreover, neither the new agent nor a member of the group is required to know about the environment and so they share the same culture in an implicit way.

Implicit Culture is the relation between a set and a group of agents such that the elements of the set behave according to the culture of the group. In the following, we informally introduce this notion, while in Appendix A, we restate the formal definition of Implicit Culture presented in [2, 3].

We assume that the agents perceive and act in an environment composed of objects and other agents. ¿From this perspective, agents are viewed as objects that are able to perceive, act and know (as a consequence of perception). Actions have as arguments objects, agents or both objects and agents. `Offer(service-1)`, `look_for(broker)` and `request(broker-1,service-1)` are examples of the three different cases, respectively. Before executing an action, an agent is faced with a "scene", which is composed of part of the environment (namely objects and agents) and the set of possible actions the agent can perform on the environment. As a particular case, the agent can also be part of the scene if reflexive actions are possible. For example, an agent `requester-1` could be faced with the environment subset {`broker-1, broker-2, service-1, service-2`} and might be able to perform the actions `request(broker-1,service-1)`, `request (broker-1,service-2)`, `request(broker-2,service-1)`, and `request(broker-2,service-2)`. Hence, an agent executes an action in a given situation (being faced with a given scene at a given time) so we say the agent executes situated actions. The agent `requester-1`,for instance, might execute the action `request(broker-2,service-1)` in a situation where he was facing the scene composed of `broker-1`, `broker-2`, `service-1`, `service-2`.

After each situated action has been executed, the agent faces a new scene. At a given time this new scene depends on the environment and on the previous action executed. If `requester-1` performs `request(broker-2,service-1)`, the `requester-1` will have the scene it faces changed because `broker-2` is now busy and not available to perform any service.

The situated executed action that an agent chooses to execute depends on its private states and, in general, it is not deteministically predictable with the information available externally. Rather, we assume it can be characterized in terms of probability and expectations. As an example, given a `requester` facing a scene in which it can perform `request(the-best-broker,service-1)`, `request(the-worst-broker, service-1)` the expected situated action can be `request(the-best-broker,service-1)`.

Given a group of agents let us suppose that there exists a theory regarding their expected situated actions. If the theory is consistent with the actions executed by the group, it can be considered a cultural constraint for the group. The theory captures the knowledge and skills of the members regarding their environment. For instance:

$$\begin{aligned}&\exists \mathtt{y} \in \mathtt{Group}:\\&\quad \forall \mathtt{x} \in \mathtt{Group}, \mathtt{s} \in \mathtt{Services}, \mathtt{b} \in \mathtt{Brokers}\\&\quad \mathtt{request}(\mathtt{x},\mathtt{y},\mathtt{s}) \wedge \mathtt{inform}(\mathtt{y},\mathtt{x},\mathtt{b}) \rightarrow \mathtt{request}(\mathtt{x},\mathtt{b},\mathtt{s})\end{aligned} \tag{1}$$

expresses that, there exists an agent `y` in the group such that, if a requester `x` requests information of `y` regarding the agents capable of performing a service `s` and `y` replies informing `x` about the broker `b`, then `x` will request `b` to perform `s`. This means that the agent `y` is able to suggest a service provider but also that such a recommendation influences the agents preferences; i.e. `x` decides to request `s` of `b`.

If the set of new agents all perform actions that satisfy the cultural constraints of the group (we call them cultural actions w.r.t. the group), then the problem of the sets suboptimal behavior with respect to that of the group is solved. We describe Implicit Culture as a relation between a set of agents G' and a group of agents G such that the agents of G' perform actions that satisfy a cultural constraint for G. When a set and a group of agents are in an Implicit Culture relation, we have an Implicit Culture phenomenon. As a consequence of an Implicit Culture phenomenon, the actions of a new `requester` of a service will be far more effective if its view of the environment includes only those `brokers` that have previously been requested for the same service.

3 A SICS for Multi-agent interaction support

Producing an Implicit Culture Phenomenon is the goal of a System for Implicit Culture Support (SICS) as presented in [2], where a general SICS architecture was introduced. In this section, we present a specific SICS architecture aimed to support multi-agent interaction.

A general SICS (see Figure 1-a) consists of three components: an observer, an inductive module and a composer. The observer stores the executed situated actions of a group of agents G in oder to make them available for the other components. The inductive module uses these actions to produce a cultural constraint theory $\mathbf{\Sigma}$ for G. Finally, the composer, using the theory $\mathbf{\Sigma}$ and the actions, manipulates the scenes faced by a set of agents G' in such a way that their expected situated actions are in fact cultural actions w.r.t 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. In a multi-agent system, a SICS can be either a general capability of the overall system or a specific capability of a single agent. In the former case, the SICS observes all the agents acting in the

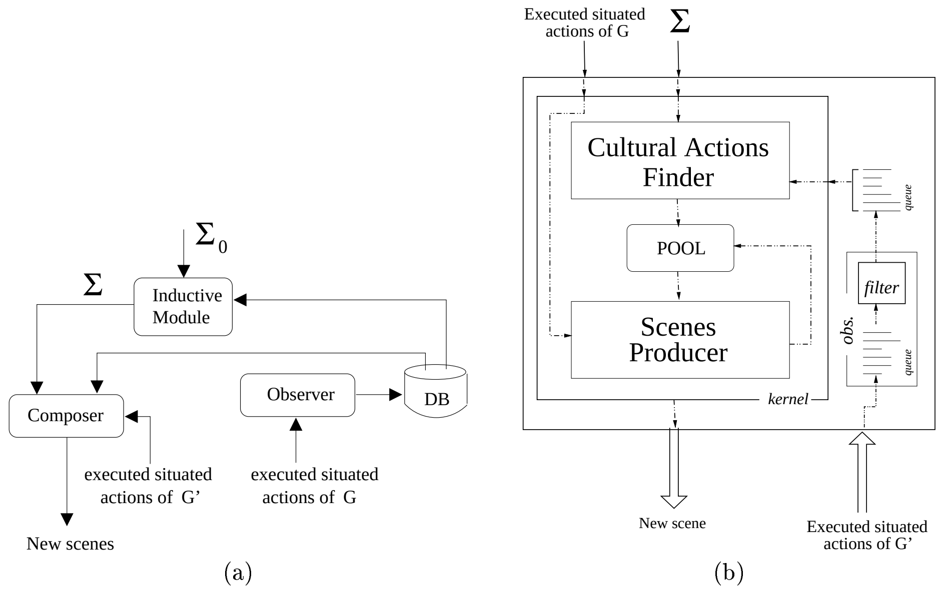

Fig. 1. (a) general architecture for SICS; (b) the composer in detail

system and manipulates the environment. In the latter, the SICS is applied to what the agent is able to observe and change, namely the part of environment and the agents it interacts with by means of actions and communicative acts. The SICS capability, both general and specific, affects the whole system. In the system we present here, we choose to adopt the second option in which a SICS is a capability of a single agent. In order to gain effectiveness we embedded a SICS in a Directory Facilitator (DF), namely an agent that plays a central role in the interactions. In particular, we adopt the idea of DF from FIPA specifications [7] and we extends its capabilities with the SICS.

Following FIPA specifications, a DF is a mandatory agent for an agent platform and provides a yellow pages directory service for other agents on the platform. Every agent that wishes to publicize its services to other agents, requests registration of the service with the DF by providing a description of these services. An agent can query the DF requesting information about the services available and the agents registered for providing such services. A DF does not guarantee the validity of the information provided in response to a request, and it does not do any kind of filtering over the information.

By means of a SICS, the DF can produce the Implicit Culture relation between an agent and the agents that have previously requested information, and provide information that influences the preferences of the agent. As presented in [2], this is a generalization of Collaborative Filtering [8, 9], where the similarity among agents is elaborated on the executed actions and not only on ratings. The presence of a SICS decreases the quantity of useless requests, thus improving the overall interaction.

We fully implemented the SICS architecture in Java language using XML for expressing the cultural constraint theory.

Architecture and cultural constraint theory
The SICS embedded in the DF is a particular case of the general one. In this specific case, we do not need any kind of theory induction over the observations, the cultural constraint theory is completely specified and the inductive module is omitted (in Figure 1-a, $\mathbf{\Sigma} \equiv \mathbf{\Sigma}_0$). The cultural constraint theory is very simple. Indeed, we want the DF to recommend agents whose services satisfy the request, namely that the expected situated action of the agent is to accept the recommendation of the DF. In other words, the agent who receives a reccomendation, will request the services of the recommended agents. Referring to the formula (1), with the agent $\mathtt{y}$ specified by the DF, the theory is:

$$\begin{aligned} &\forall \mathtt{x} \in \mathtt{Group}, \mathtt{s} \in \mathtt{Services}, \mathtt{b} \in \mathtt{Brokers} \\ &\quad \mathtt{request}(\mathtt{x}, \mathtt{DF}, \mathtt{s}) \wedge \mathtt{inform}(\mathtt{DF}, \mathtt{x}, \mathtt{b}) \rightarrow \mathtt{request}(\mathtt{x}, \mathtt{b}, \mathtt{s}) \end{aligned} \tag{2}$$

In general, our architecture accepts cultural theories 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". Antecedent and consequent are conjunctions of atoms, namely two types of predicates: observations on an agent and conditions on times. For instance, $\mathtt{request(x,y,s,t_1)}$ is a predicate of the first type that says that the agent $\mathtt{x}$ requests of agent $\mathtt{y}$ the service $\mathtt{s}$ at the time $\mathtt{t}_1$; while $\mathtt{less(t_1,t_2)}$ is an example of the second type and it simply states that $\mathtt{t_1 < t_2}$. The following rule is used to express the cultural theory (2):

$$\begin{aligned} &\mathtt{request}(\mathtt{x}, \mathtt{DF}, \mathtt{s}, \mathtt{t}_1) \wedge \mathtt{inform}(\mathtt{DF}, \mathtt{x}, \mathtt{y}, \mathtt{t}_2) \wedge \mathtt{less}(\mathtt{t}_1, \mathtt{t}_2) \rightarrow \\ &\quad \mathtt{request}(\mathtt{x}, \mathtt{y}, \mathtt{s}, \mathtt{t}_3) \wedge \mathtt{less}(\mathtt{t}_2, \mathtt{t}_3) \end{aligned} \tag{3}$$

which states that if $\mathtt{x}$ asks the DF for the service $\mathtt{s}$, and the DF replys informing $\mathtt{x}$ that $\mathtt{y}$ can provide such a service, then $\mathtt{x}$ will request of $\mathtt{y}$ the service $\mathtt{s}$.

Observer
The observer observes and stores the actions performed by the agents interacting with the DF. In particular, it stores the requests an agent sends to the DF, the responses that the DF sends to the agent, and the subsequent requests for the service sent by the requesting agent to the recommended service providing agent. The information stored is of the form:

- $\mathtt{request(x,y,s,t)}$, $\mathtt{x}$ sends a request to $\mathtt{y}$ for the service $\mathtt{s}$ at time $\mathtt{t}$;
- $\mathtt{inform(x,y,o,t)}$, $\mathtt{x}$ informs $\mathtt{y}$ about $\mathtt{o}$ at time $\mathtt{t}$.

Composer
The goal of the composer is propose a set of scenes to agents of G' such that the

```
loop
  get the last executed situated action α
  for all rule r of Σ do
    for all atom a of r's antecedent ant do
      if match(a,α) then
        if find-set(ant,past-actions) then
          join(past-actions,r)
          return r's consequent cons
        end if
      end if
    end for
  end for
  return false
end loop
```

Fig. 2. The algorithm for the CAF submodule

expected situated actions of these agents satisfy the cultural constraint theory $\mathbf{\Sigma}$ for the group G. In our case, G' is composed of only one agent (namely the one currently requesting information for the DF), while G is composed of all the agents that have interacted with the DF in the past. The cultural constraint $\mathbf{\Sigma}$ is expressed by the rule (3) and the scenes are composed of the agents recommended by the DF. Figure 1-b shows the composer in detail.

Basically, the composer consists of two main submodules: [1]

- the *Cultural Actions Finder* (CAF), which 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), which 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.

When an agent `x` asks the DF for a service `s` (i.e. performs the action `request(x, DF,s,t`$_1$`)`) the CAF using (3) finds the cultural action `request(x,y,s,t`$_2$`)`, with `t`$_2$`>t`$_1$, and the SP proposes an agent `y`, of which `x` is expected to request the service `s`.

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 2 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` matches with the executed situated action α; then the function *find-set*(`ant,past-actions`) finds a set `past-actions` of past

[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.

executed situated actions that matches with the set of atoms of `ant`; and finally, the function *join*(`past-actions,r`) joins the variables of `r` with the situated executed actions in `past-actions`.

Scenes Producer
Given a cultural action α for the agent x (e.g., `request(x,y,s,t`$_2$`)`), the algorithm used in SP consists of three steps:

1. find a set of agents Q that have performed actions similar to α;
2. select a set of agents $Q' \subseteq Q$ similar to x and the set of scenes S in which they have performed the actions;
3. select and propose to x a scene from S.

Steps 1 and 2 are based on two domain dependent similarity and selection functions, and we omit the details. Step 3 selects the scenes in which the cultural action is the expected situated action. To make this selection we first estimate the similarity value between the expected action and the cultural action for each scene $s \in S$, and then we select the scene with the maximum value. The function to be maximized here is the expected value $E(sim(\beta_x, \alpha)|s)$, where β_x is the action performed by the agent x, α is the cultural action, and $s \in S$ is the scene in which β_x is situated. $sim(\beta_x, \alpha)$ is a domain–dependent similarity function and in our case we have chosen:

$$sim(\beta, \alpha) = \begin{cases} 1 & if\ \beta = \alpha \\ 0 & otherwise \end{cases}$$

where $\beta = \alpha$ if the actions are of the same type (namely, `request` or `inform`) and have the same arguments. The following formula is used to estimate E:

$$\hat{E}\left(sim(\beta_x, \alpha)|s\right) = \frac{\sum_{u \in Q'} \hat{E}_1\left(sim(\beta_u, \alpha)|s\right) * w_{x,u}}{\sum_{u \in Q'} w_{x,u}} \tag{4}$$

that is we calculate the weighted average of the similarity value for the expected actions of neighboring scenes. $w_{x,u}$ is the similarity between the agent x and the agent u, while E_1 is estimated as follows:

$$\hat{E}_1\left(sim(\beta_u, \alpha)|s\right) = \frac{\sum_{i=1}^{m} sim(\beta_u^i, \alpha)}{m} \tag{5}$$

that is the average of $sim(\beta_u^i, \alpha)$ over the m actions performed by u in s.

4 The eCulture Brokering System

In this section, we present the eCulture Brokering System, a multi-agent system for brokering cultural information, and to which we have applied the SICS presented above. The multi-agent system has been developed using JACK Intelligent Agents, an agent-oriented development environment built on top of and

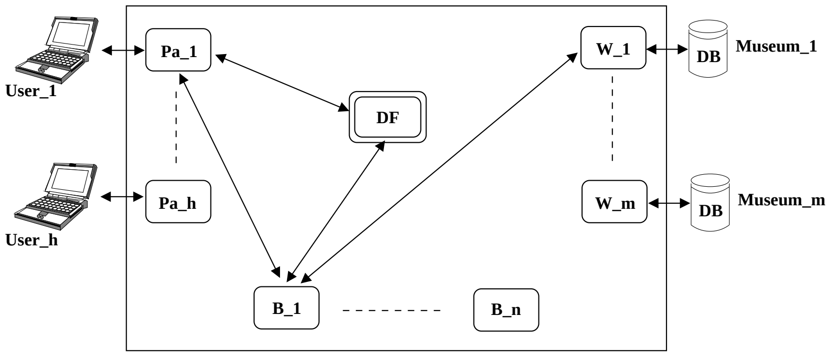

Fig. 3. The eCulture Brokering System

fully integrated with the Java programming language [4]. The system is the result of a collaboration between ITC-irst and University of Trento. In the same context Busetta et. al [5, 1] have developed the basic idea of the implicit support of the SICS architecture presented in [2] for the particular case of messages exchange between agents. They have introduced the innovative overhearing architecture, in which a listener agent observes the messages that two or more parties exchange and a suggester agent composes suggestions and sends them to agents in order to reach the goals fixed *a priori*.

Figure 3 shows a part of the architecture of the eCulture Brokering System (CBS). In particular, it focuses on the agent interaction for which we use the SICS. The platform contains a set of personal agents `Pa_1`,...,`Pa_h`, a `DF` that provides information about a set of brokers `B_1`,...,`B_n`, and a set of wrappers `W_1`,...,`W_m`. A personal agent is created and assigned to each user who accesses the system by means of a web browser. The brokers are specialized in providing information about a specific cultural area (e.g. history, archeology, art, etc...), and they can collect information from different wrappers. Each wrapper is built for a particular database implementation used by a museum. Basically, the databases are of two types: Microsoft Access and Oracle. The complete architecture includes other agents, like for instance the agent resource broker, which provides information about the resources available outside the multi-agent system.

Let us consider a typical interaction. The user can ask the personal agent to search for cultural information about a specific century. The personal agent sends a request to the `DF` for the most appropriate broker that may satisfy the user. The `DF` replys to the personal agent with the name of a broker, and waits for a response message stating whether the personal agent is satisfied with the recommended broker. In case of a negative response, the DF sends the name of another broker to the personal agent and waits again. Having received the name of a broker from the `DF`, the personal agent sends a request to this broker, which in turn asks one or more wrappers to retrieve information from the associated databases.

Agent	B_1	B_2	B_3	B_4	B_5	B_6
Pa_1	s(XVI)	s(XVI)	u(XVI)	s(VI)	s(IV)	s(VI)
Pa_2	s(XVI)		u(XVI)	s(VI)		
Pa_3	s(VI) u(XVI)	s(XVI)	u(XVI)		u(IV)	u(VI)
Pa_4	s(XVI)		u(XVI)			

Table 1. Effects of a simple interaction on the state of the DF

The DF uses the SICS to decide which broker to suggest to the personal agent. In particular, for each personal agent that sends a request, the DF finds all the agents that have previously performed similar actions (requests and consequent response messages), and then suggests to the personal agent a particular broker with which such similar agents would be satisfied.

Table 1 reports the results of a simple set of interactions between the DF and four personal agents (Pa_1,Pa_2,Pa_3, and Pa_4). In each interaction the personal agent has requested the name of a broker for a specific century (for instance, IV, VI, XVI, ...), and the DF has stored whether its suggestion has satisfied the personal agent (s for satisfied, and u for unsatisfied). Let us suppose that in this situation the personal agent Pa_4 asks the DF for a broker about the VI century. The DF looking at the table finds that agents Pa_1 and Pa_2 are similar to Pa_4, and suggests Pa_4 with the broker that should satisfy Pa_1 and Pa_2, namely with B_4. In case that Pa_4 is not satisfied, the DF will recommend B_6, which has satisfied Pa_1. Of course, this is a very simple case in which it is possible to see which are the similar agents, without any particular elaboration. However, our system is intended to work in complex situations with hundreds of personal agents and, for each of them, tens of requests.

The experiments we have made using the CBS have shown that the SICS can effectively improve the interaction among agents. In particular, it can help new agents (users), that do not know the domain, to interact with the multi-agent system.

5 Conclusion

In this paper, we have presented the concept of Implicit Culture and we have shown how to use a System for Implicit Culture Support for supporting multi-agent interaction. We have presented an implementation of SICS and the eCulture Brokering Systems, a multi-agent system in which the SICS is used.

Implicit Culture Support allows us to improve the interaction among agents without need to equip the agents with additional capabilities. Extending the use of SICS to other agents, such as for instance the agent resource broker (which provides information about the resources available outside the multi-agent system), and implementing the inductive module for inducing cultural constraint theories for different group of agents, are the objectives of our future work.

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 charateristics 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:

$$F_{\mathcal{E}} : A \times \mathcal{S}_{\mathcal{E},\mathcal{A}} \times T \to \mathcal{S}_{\mathcal{E},\mathcal{A}} \tag{6}$$

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*

$$\{E(h_{b,t}|\langle b, \sigma, t \rangle) = \alpha\}, \mathbf{\Sigma} \not\vdash \perp$$

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.

References

1. M. Aiello, P. Busetta, A. Donà, and L. Serafini. Ontological overhearing. In *Proc. of 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001), Seattle, USA*, 2001.
2. E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture. In *Proc. of the Workshop on Agents and Reccomender Systems, in Agents2000*, Barcellona (http://www.science.unitn.it/~pgiorgio/ic), 2000.
3. E. Blanzieri, P. Giorgini, and F. Giunchiglia. Implicit culture and multi-agent systems. In *Proc. of the Int. Conf. on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, L'Aquila - Italy (http://www.science.unitn.it/~pgiorgio/ic), 2000.

4. P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - components for intelligent agents in java. Technical Report TR9901, AOSs, January. http://www.jackagents.com/pdf/tr9901.pdf.
5. P. Busetta, L. Serafini, D. Singh, and F. Zini. Extending multi-agent cooperation by overhearing. In *Proc. the Sixth International Conference on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, 2001.
6. K. Decker, K. Sycara, and M. Williamson. Matchmaking and brokering. In *Proc. of the Second International Conference on Multi-agents Systems (ICMAS-96)*, 1996.
7. FIPA. *Foundation for Intelligent Physical Agents.* http://www.fipa.org.
8. D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
9. J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
10. K. Sycara and D. Zeng. Multi-agent integration of information gathering and decision support. In *Proceedings of the 12th European Conference on Artificial Intelligence*, John Wiley & Sons, Ltd., 1996.
11. G. Weiss. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, 1999.
12. M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.