Implicit Culture for Information Agents

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

[1] ITC-irst Trento - Italy
[2] Department of Psychology - University of Turin - Italy
[3] Department of Mathematics - University of Trento - Italy
pgiorgini@science.unitn.it

Abstract. Earlier work introduced the concept of Implicit Culture and its use in multi-agent systems. Implicit culture support can be seen as a generalization of Collaborative Filtering and it can improve agents' performances. In this paper, we present an implementation of a System for Implicit Culture Support, results obtained in a recommendation-problem domain, and an application to the eCulture Brokering System, a multi-agent system aimed to mediate the access to cultural information.

1 Introduction

Given the problem of information overload, the development of information agents is a mayor issue [11]. On the other hand, Collaborative Filtering demonstrated to be an effective solution to the problem of information overload, specifically for products recommendation. In [1], we have introduced the notion of Implicit Culture, in which the application of the collaborative filtering approach to agent-based systems extends the performances of information agents.

Systems for Implicit Culture Support (SICS in the following) have the goal of estabilishing an Implicit Culture phenomenon. Informally, Implicit Culture is 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. 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. This concept can help the analysis of existing systems and suggest ideas for the synthesis of new ones. In fact, support of Implicit Culture can be useful for various applications: Computer Supported Collaborative Work, Group profiling, Cognitive modelling of social phenomena, e-books, Computer Mediated Communication Systems and, as we briefly present in the following, generalization of collaborative filtering, Knowledge Managment and requirements and interaction control of agent-based systems.

Collaborative filtering [8, 9, 12, 17] is a popular technique exploited in recommendation systems. The goal is information filtering, namely to extract from a usually long list of items, e.g. links or products, a little set that the user could prefer. Collaborative filtering exploits correlations in the pattern of preferences expressed actively o passively by other users in terms of ratings. Differently from the content-based filtering, collaborative filtering does not rely on the content or

shape of objects. The central idea is to automate the process of recommending items to a user on the base of the opinions of people with similar preferences. As suggested by Blanzieri and Giorgini [1] collaborative filtering can be seen as a System for Implicit Culture Support (SICS).

In Knowledge Management, generally 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 abilities of the members of a group of people. In [14], 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, they conceive it as something that transforms it. Implicit Culture can be applied successfully in this context. In particular, 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.

Advantages of Implicit Culture support has been proposed also for artificial agents [6, 2], in particular for controlling the requirements of agent-based systems and supporting their interaction. Autonomy of agents, unknown properties of the environment and insertion of new agents do not allow to foresee completely the multi-agent system's behavior in the modeling phase. As a consequence, the overall system can fail to fulfill the desired requirements. In particular, requirements should persist after a composition changing of the group of agents, that is the new agents should act consistently with the culture of the group. Using SICSs, it is possible to modify the view that the agents have of the environment and, consequently, change the set of possible actions that the agents can perform in the environment. Working on the possible actions, a SICS is able to lead the new agents to act consistently with the behavior of the group.

The architecture of SICS proposed in [1, 6] relies on the exploitation of learning techniques. In particular it is possible to identify two learning problems: (i) induction of a cultural theory on the behavior patterns of the group and (ii) prediction of a scene such that the elements of the set will behave consistently with the cultural theory. The first problem can be solved by standard data mining techniques. In this paper we present an implemented SICS that solves the second problem exploiting an original generalization of a memory-based Collaborative Filtering algorithm. The SICS is applied to two different information access problems.

The paper is organized as follows. Section 2 presents the SICS architecture and the learning problems. In Section 3 we show a solution to the problem of prediction of the scene and in Sections 4 and 5 we present some experimental results and a real-world application, respectively. The final section hosts conclusions and future work.

2 Systems Implicit Culture Support

The goal of a SICS is to establish an implicit culture phenomenon. In the following, we informally introduce the notions of implicit culture and implicit culture phenomenon (Appendix A reports the formal definitions given in [2]). The second part of the section presents the general architecture of a SICS and shows how it relies on learning techniques.

An Implicit Culture phenomenon is a pair composed by a set and a group of agents such that the elements of the set behave according to the culture of the group and Implicit Culture is the relation between the elements of the pair. The definitions are expressed in terms of expected situated actions and cultural constraint theories. Some assumptions underlie these concepts.

We assume that the agents perform situated actions. Agents perceive and act in an environment composed of objects and other agents. In this perspective, agents are objects that are able to perceive, act and, as a consequence of perception, know. Before executing an action, an agent faces a scene formed by a part of an environment composed of objects and agents. Hence, an agent executes an action in a given situation, namely the agent and the scene at a given time. 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.

Another assumption states that the expected situated actions of the agents can be described by a cultural constraint theory. The action that an agent executes depends on its private states and, in general, it is not deteministically predictable with the information available externally. Rather, we assume that it can be characterized in terms of probability and expectations. 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.

We call Implicit Culture a relation between a set of agents G' and a group 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 Implicit Culture relation, we have an Implicit Culture phenomenon. The definitions do not require the empirical validation of the cultural constraint theory against the executed actions of G.

The definition of Implicit Culture (Appendix A briefly discusses the use of "implicit") does not give sufficient conditions for its realization, posing the problem of its support in practise. The general architecture of a SICS proposed in [1] allows to achieve the goal of estabilishing an implicit culture phenomenon following two steps. First, the elaboration of a cultural constraint theory $\mathbf{\Sigma}$ from a given domain and a set of situated executed actions of a group G. Second, the proposal to a group G' of a set of scenes such that the expected situated actions of the set of agents G' satisfies $\mathbf{\Sigma}$. Both the steps present learning problems.

A general SICS (see Figure 1-a) consists of three components: observer, inductive module and composer. The observer stores the situated executed 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

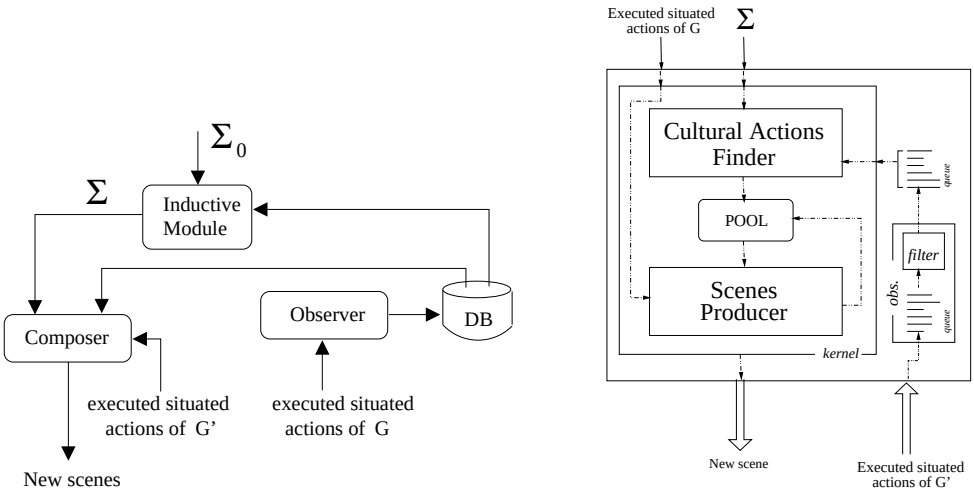

Fig. 1. Architecture (a) The composer in detail(b)

$\mathbf{\Sigma}$ 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 cultural action w.r.t G. As a result, the agents of G' executes (on average) cultural actions w.r.t G (implicit culture phenomenon).

In Figure 1-a the composer proposes to the agents a, b, and c the scenes σ_{t+1}, σ'_{t+1}, and σ''_{t+1}, respectively. Notice that in this case the agents b and c belong to both G and G'. This means that also their situated actions are stored in DB and thus they are used to elaborate the theory $\mathbf{\Sigma}$ and the new scenes.

In general, our implemented 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 said antecedent and $C_1 \wedge \cdots \wedge C_m$ 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, `request(x,y,s,`$\mathtt{t}_1$`)` is a predicate of the first type, that says that the agent `x` requests the agent `y` for the service `s` at time $\mathtt{t}_1$; whereas `less(`$\mathtt{t}_1$`,`$\mathtt{t}_2$`)` is an example of the second type and it simply says that $\mathtt{t}_1 < \mathtt{t}_2$.

The composer proposes to a set of agents G' a set of scenes such that their expected situated actions satisfy a cultural constraint theory $\mathbf{\Sigma}$ for a group G. The main idea is splitting the problem in two sub-problems: (i) find the cultural actions and (ii) find the scenes where such actions are the expected situated actions. Figure 1-b shows the composer in detail. Basically, the composer consists of two main submodules and an additional component:

- 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 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.
- the *Scenes Producer* (SP), that takes one of the cultural action produced by the CAF and, using the executed situated actions of G, produces scenes such the expected situated action is the cultural action.
- the *Pool*, an additional component, 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.

The SICS architecture requires the solution of two learning problems. A problem of induction of the cultural constraint theory (Inductive Module) and a problem of prediction of scenes (Scenes Producer).

Inductive Module Problem. Given a set of situated executed actions performed by the agents of G, find a cultural constraint theory.

Scene Producer Problem. Given a set of situated executed actions of the agents of G and G', and given a cultural action α for the agent x, find a scene s such that the expected situated action of x in the scene s is α.

The Inductive Module Problem is a rather standard learning problem: inducing the behavior patterns of a group and it is possible to solve using standard data mining techniques. As we previously noted, we do not require any specific validation of the theory. Obviously, very different effects will be reached depending on the fact that the theory is validated or not. In the Scene Producer Problem the request is on the effectiveness of the scene w.r.t the goal of producing the execution of a given action, namely its *persuasiveness*.

3 A Solution to the Scene Producer Problem

The solution exploits the principles of instance-based learning (namely, memory-based or lazy). Given a cultural action α for the agent x 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 that 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 performed actions;
3. select and propose to x a scene of S.

Figure 2 shows the 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. Sections 4 and 5 contain two examples of similarity function (eq. 5 and 7).

```
for all y ∈ G'
    for all situated executed actions β_y of y
        if sim(β_y, α)> T_min then {
            if y ∉ Q then  y → Q
            s → S(y)
        }
```

Fig. 2. The algorithm for step 1

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

$$w_{x,y} = \frac{1}{|S_{xy}|} \sum_{s \in S_{xy}} \frac{1}{N_x(s) N_y(s)} \sum_{\beta_x \in N_x(s)} \sum_{\beta_y \in N_y(s)} sim(\beta_x, \beta_y) \tag{1}$$

where $S_{xy} = S(x) \cap S(y)$ is the set of scenes in which both x and y have executed at least an action. $N_x(s)$ and $N_y(s)$ are the set of actions that x and y have respectively performed in the scene s. Eq. 1 can be replaced by a domain-dependent agent similarity function if needed (e.g., Eq. 4 in Section 4).

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

$$\hat{E}\left(sim(\beta_x, \alpha)|s\right) = \frac{\sum_{u \in Q'} \hat{E}_1\left(sim(\beta_u, \alpha)|s\right) * w_{x,u}}{\sum_{u \in Q'} w_{x,u}} \tag{2}$$

that is we calculate the weighted average of the similarity of the expected actions for the neighbor of the scene, $w_{x,u}$ is the similarity between the agent x and the agent u, whereas E_1 is estimate as follows:

$$\hat{E}_1\left(sim(\beta_u, \alpha)|s\right) = \frac{1}{|N_u(s)|} \sum_{\beta_u \in N_u(s)} sim(\beta_u, \alpha) \tag{3}$$

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

The algorithms described above and the general architecture described in Section 2 are fully implemented in Java using XML for expressing the cultural constraint theory.

4 Experiments in a recommendation-problem domain

Collaborative filtering can be seen as a particular SICS. In this section we present the results of some experiments aimed to compare collaborative filtering and the

SICS parametrized for a recommendation problem. The goal of the experiments is validating the system against a well-estabilished method on a particular domain.

For collaborative filtering we use a memory and neighborhood based algorithm presented by Herlocker et al. in [9]. The algorithm consists of three basic steps. In the first step all the users are weighted with respect to their similarity with the active user by *Pearson correlation coefficient*:

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a) * (r_{u,i} - \overline{r}_u)}{\sigma_a * \sigma_u} \tag{4}$$

where m is the number of objects co-valutated, $r_{a,i}$ is the ranking given by the user a to the object i, $\overline{r}_a$ and σ_a are respectively the average and variance of the rankings of a. In the second step the best k_{CF} correlates are picked in order to compute a prediction (third step) using the *deviation-from-mean* approach introduced in [17].

The SICS used for the experiments is characterized by: a set of agents (users) $\mathcal{P} = \{u_1, \cdots, u_n\}$; a set of objects $\mathcal{O} = M \cup V$ (M is the set of items and $V = \{0, 0.2, \cdots, 0.8, 1\}$ the set of the possible votes) and a set of possible actions $\mathcal{A} = \{vote, request\}$. For a recommender system the cultural theory is specified in advance so no inductive module is needed:

$$\forall x \in P, m \in M: \quad request(x) \rightarrow vote(x, m, v_{max})$$

is expressed by the following rule:

$$request(x, t_1) \rightarrow vote(x, m, v_{max}, t_2) \wedge less(t_1, t_2)$$

that states that if x requests a suggestion, then x will assign the maximum vote to the proposed item. For a recommender system we require the satisfation of the user with the recommended items. For similarity between actions we use the following domain-dependent function:

$$sim(A_1, A_2) = \begin{cases} 0 & if(A_1 = vote \wedge A_2 = request) \vee \\ & (A_1 = request \wedge A_2 = vote) \vee \\ & (A_1 = vote(x, o, v_1) \wedge \\ & A_2 = vote(y, p, v_2) \wedge o \neq p) \\ 1 & if(A_1 = A_2 = request) \\ 1 - |v_1 - v_2| & if(A_1 = vote(x, o, v_1) \wedge \\ & A_2 = vote(y, p, v_2) \wedge o = p) \end{cases} \tag{5}$$

that means that the similarity is zero when the actions are different or when they are both *vote* but about distinct objects; it is maximum (namely 1) when the actions are both *request*; and finally, it is $1 - |v_1 - v_2|$ when the actions are both *vote* about the same object, but with different numerical votes.

For the experimentation we used the database EachMovie [13] that collects data of 72961 users who voted 1623 movies. We built a dataset considering the first 50 movies and the 119 users with at least one vote among the first 300 and we run a leave-on-out w.r.t the users. The CF algorithms returns a list of estimated ranking while in this case the SICS returns only a scene composed

action sim	eq. 5 (domain-dependent similarity)								eq 7
agent sim	eq. 4 (Pearson Correlation coefficient)							eq. 1	eq. 4
sim min	T_{min}=0						T_{min}=0.8	T_{min}=0	T_{min}=0
neighbors	k=10	k=30	k=50	k=80	k=150	k=300	k=80	k=80	k=80
ranks									
1	25.4	57.6	65.3	68.6	68.6	69.5	66.1	58.0	6.8
2	14.4	11.0	11.0	10.2	10.2	11.0	12.7	14.3	4.2
3	8.5	7.6	6.8	7.6	8.5	6.8	6.8	8.4	4.2
4	10.2	5.1	5.1	4.2	4.2	3.4	5.9	5.9	4.2
5	6.0	2.5	1.7	0.0	0.0	1.7	0.0	1.7	5.1
over 5	35.5	16.2	10.1	9.4	8.5	7.6	8.5	11.7	75.5

Table 1. Experimental results. Distribution in percentage of the items proposed by SICS in the rankings position proposed by the CF algorithm with $K_{CF} = 30$.

by a movie. We compared the movie proposed by the SICS and the best 10 movies ranked by the CF algorithm computing the distribution in percentage. We run our implementation of the CF algorithm obtaining a Mean Absolute Error comparable to the ones presented in the literature [3, 7]. We run the experiments on the SICS changing the number of neighbours, the threshold and the functions used to compute the similarity among agents and actions. Table 1 presents the results of the experiments.

A low k implies a low number of scenes (items) valutated and consequently low performance (compare columns 1-6) because it is likely to miss some valuable items. With $T_{min} = 0$ the number of valutated scenes is on average 38.4 against 25.2 with $T_{min} = 0.8$, hence the speed of presentation is higher with a slight decrease in the performance (compare columns 4 and 7). The domain-dependent agent similarity function eq 4 is slightly better that the general one 1 (compare columns 4 and 8). Finally, a very simple action similarity (eq. 7) performs poorly (compare columns 4 and 9). We can conclude that with domain-dependent similarities our SICS is comparable with CF. More interestingly, comparing the last cells of column 4 and 8 it is possible to conclude that also the version without the domain-dependent agent similarity but with a general one performs satisfactorily.

5 An application: The eCulture Brokering System

In this section, we present the eCulture Brokering System [2], a multi-agent system for cultural information brokering where we have applied the SICS. The multi-agent system has been developed using JACK Intelligent Agents [4].

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 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. The SICS capability, both general and specific, affects the whole system. In the system we present here, we choose to adopt the second option where 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 [5] and we extends its capabilities with the SICS.

A DF is a mandatory agent of an agent platform that provides a yellow pages directory service to agents. Every agent that wishes to publicize its service to other agents, requests the registration to the DF providing a description of its services. An agent can ask the DF in order to request information about the services available and the agents registered for such services. 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 encounters the preference of the agent.

In particular, it focus on the agents 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 (for instance, history, archeology, art, etc...), and they can collect information from different wrappers. Each wrapper is built for a specific museum's database. 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. The rule used to express the cultural theory is the following:

$$\begin{aligned} &\texttt{request}(\texttt{x}, \texttt{DF}, \texttt{s}, \texttt{t}_1) \wedge \texttt{inform}(\texttt{DF}, \texttt{x}, \texttt{y}, \texttt{t}_2) \wedge \texttt{less}(\texttt{t}_1, \texttt{t}_2) \rightarrow \\ &\quad \texttt{request}(\texttt{x}, \texttt{y}, \texttt{s}, \texttt{t}_3) \wedge \texttt{less}(\texttt{t}_2, \texttt{t}_3) \end{aligned} \tag{6}$$

that states that if x asks DF for the service s, and DF replays informing x that y can provide such a service, then x will request to y the service s.

The similarity function between two actions is the following:

$$sim(\beta, \alpha) = \begin{cases} 1 & if\ \beta = \alpha \\ 0 & otherwise \end{cases} \tag{7}$$

where $\beta = \alpha$ if the actions are of the same type (namely, request or inform) and have the same arguments.

The DF uses the SICS to suggest the broker 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 the personal agent with the broker for which such agents would be satisfied. The experiments we have made 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.

6 Conclusions and future work

We have presented an implementation of a System for Implicit Culture Support that exploit an application of instance-based learning techniques. We have

showed that, in a particular domain and with a simple a priori theory, the system is functionally equivalent to Collaborative Filtering. Moreover we have presented a real-world application. Our three-steps algorithm for proposing the scene can be considered a generalization of a Collaborative Filtering algorithm, where the similarity is performed on executed actions and not on ratings. This generalization is non–trivial for it puts the SICS in a wider framework than simple CF. In fact it is possible to vary the domain, the cultural contraint theory, and also to deal with artificial agents as we have shown with the eCulture system. A relevant portion of research on multi-agents learning [18] deals with reinforcement learning (e.g., [10]). From the point of view of reinforcement learning the works of Price and Boutillier [16] on *Implicit Imitation* are relevant to our work. The critical difference is that a SICS does not imitate but *induce* an agent to imitate or more generally to act consistently with another group of agents. In the broad area of web personalization a relevant work is the one by Paliouras et. al. [15] who clusters communities of on-line users. In our perspective the groups are given, consequently an integration of the methods would be interesting. Finally, our approach can be seen as a temptative of supporting organizational learning in the direction proposed in [19]. Future work will be devoted to experimentation on domains with a wider range of actions, more complex scenes and more complex, possibly induced, cultural constraint theories.

Acknowledgments

The authors wish to thank Fausto Giunchiglia and Paolo Traverso for their precious support and Paolo Massa and Sabrina Recla for their work on the earlier phases of the project.

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:

$$F_{\mathcal{E}} : A \times \mathcal{S}_{\mathcal{E},\mathcal{A}} \times T \to \mathcal{S}_{\mathcal{E},\mathcal{A}} \tag{8}$$

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*

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

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

Definition 12 (implicit culture). Implicit Culture *is a relation $\asymp$ between two parties G and G' such that G and G' are in relation ($G \asymp G'$) 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. E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture. In *Proceedings of the Workshop on Agents and Recommender Systems*, Barcellona, 2000.
2. E. Blanzieri, P. Giorgini, P. Massa, and S. Recla. Implicit culture for multi-agent interaction support. In *Proc. of the Conf. Cooperative Information Systems COOPIS*, 2001.
3. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the Fourteenth Conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1998.
4. P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - components for intelligent agents in java. AOS TR9901, January 1999. http://www.jackagents.com/pdf/tr9901.pdf.
5. FIPA. *Foundation for Intelligent Physical Agents*. http://www.fipa.org.
6. E. Blanzieri P. Giorgini and F. Giunchiglia. Implicit culture and multi-agent systems. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, (SSGRR 2000)*, L'Aquila - Italy (http://www.science.unitn.it/ pgiorgio/ic), 2000.
7. A. Gokhale. Improvements to collaborative filters algorithms. Master's thesis, Worchester Ploytechnic Institute, May 1999.
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. Herlocker, J. Konstan, A. Borchers, and J Riedl. An algorithmic framework for performing collaborative filtering. In *Conference on Research and Development in Information Retrieval*, 1999.
10. J. Hu and M. Wellman. Multiagent reinforcement learning theoretical framework and an algorithm. In *Proceedings of the ICML*, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers.

11. M. Klusch and F. Zambonelli, editors. *Cooperative Information Agents V. 5th International Workshop, CIA 2001, Modena, Italy, September 6-8, 2001. Proceedings*, volume 2182 of *LNCS*. Springer-Verlag, 2001.
12. 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.
13. P. McJones. Eachmovie collaborative filtering data set, dec systems research center, 1997. http://research.compaq.com/SRC/eachmovie/.
14. I. Nonaka and H. Takeuchi. *The knowledge Creating Company*. Oxford University Press, New York, 1995.
15. G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. Spyropoulos. Clustering the users of large web sites into communities. In *Proceedings of the ICML*, Stanford, 2000. Morgan Kaufmann Publishers.
16. B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proceedings of the ICML*, Bled, 1999. Morgan Kaufmann Publishers.
17. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, 1994.
18. P. Stone and M. Veloso. Multiagent systems: A survey from amachine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.
19. K. Takadama, T. Terano, and K. Shimohara. Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents*, 1999.