An Adaptive and Socially-Compliant Trust Management System for Virtual Communities

Reda Yaich, Olivier Boissier, Philippe Jaillon, Gauthier Picard
Ecole Nationale Supérieure des Mines de Saint-Étienne
Saint-Étienne, France
{firstname.name}@emse.fr

ABSTRACT

Virtual communities (VCs) are open socio-technical structures wherein autonomous entities (i.e. agents) with common objectives join together to mutually satisfy their goals. The success of these communities relies on collaboration and resource sharing principals, making trust a critical issue for each member. Such environments motivate the need for more flexible trust models wherein both individual (i.e. user-centred) and collective (i.e. community-centred) trust requirements are considered in the decision making-process.

This paper reports our on-going efforts in that perspective and presents our Adaptive and Socially-Compliant Trust Management System (ASC-TMS). Policies are used, in the system, to specify individual and collective trust requirements, while meta-policies enable agents to dynamically adapt their policies and make socially-compliant trust decisions through automatic combination of individual and collective policies.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination, Multiagent systems, Intelligent agents; K.6 [**Management of Computing and Information Systems**]: System Management, Security and Protection

General Terms

Trust, Policies, Trust Management Systems, Collaboration

Keywords

Trust Management, Virtual Communities, Social Influence, Open Innovation, Agents

1. INTRODUCTION

The last decade have seen an ever-growing impact of trust in the decision-making process within online environments. Alongside, numerous Trust Management Systems (TMS) [3, 5, 21] have been proposed to enable distributed specification and assessment of trust requirements using policies. Policies constitute a flexible way to express under which conditions (i.e. based on which factors) trust decisions can be made [21]. However, the advent of Virtual Communities introduced new issues that are challenging existing TMS on how to specify and manage trust policies [13].

One of the main issues when specifying and enforcing trust policies within open, dynamic and evolving environments like VCs is the ability of these policies to handle unforeseen situations. Such adaptation capability is essential. Otherwise, running policies could become rapidly irrelevant or inefficient [8, 13]. Another important issue, when deploying trust policies in virtual communities, is the capability of trust management systems to combine individual and collective policies during the trust assessment [4, 10, 17]. The consideration of collective trust requirements imposed by the community has been recognized as an important factor of social integration in the *social influence theory* [4].

To that aim, we propose a policy-based [5,21] trust model wherein policies are considered as concrete implementations of trust requirements, meta-policies are used to automatically adapt policies, and where individual policies are combined with collective ones to yield aggregated trust requirements. To implement and illustrate desired aspects of the model, we used Jason [7], a multi-agent oriented programming platform, which allowed us to address the issues discussed earlier while preserving the generality of the approach.

The rest of the paper proceeds as follows. In the next Section we motivate our approach using an *Open Innovation* community scenario wherein requirements for trust management has been identified. In Section 3, we formally introduce the system model, basic foundations and concepts our proposal is built on. In Section 4 we present the proposed trust model outline and its related concepts. The Section 5 we describe how trust policies are specified, adapted, combined and evaluated. Finally, we briefly survey related works, discuss our contribution and conclude with some perspectives in Section 6.

2. MOTIVATIONS

In this Section, we first introduce a motivating scenario for trust management in virtual communities. Then we use this scenario to derive and motivate some important features the proposed Trust Management System should integrate.

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

2.1 Motivating Scenario

Open Innovation [2] is currently recognized as a new model that companies and organizations are adopting to enhance innovation in their R&D departments by harnessing external ideas and stimulating creativity. Based on this concept, a number of commercial (e.g. Hypios.com) and free (e.g. W3C Community Group) platforms have been proposed to facilitate the innovation process and the creation of Innovation Communities (IC). These platforms constitute a showcase for R&D problems to which the community members try to find a solution and earn money. Members can try to solve individually proposed problems, but in most of the cases they join together and create a dynamic community wherein problems are solved collaboratively. The success of these communities relies on full collaboration and massive resource sharing, making trust a critical issue for each member.

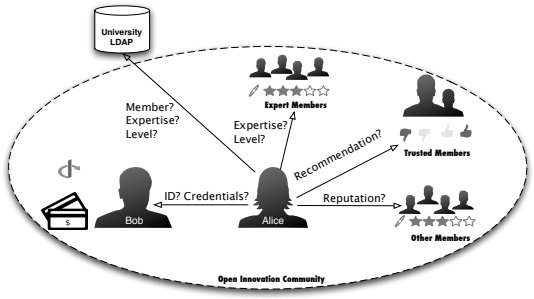

Figure 1: Trust factors and information sources diversity within Innovation Communities.

Members of these communities like *Alice* and *Bob* have a variety of information sources they can use to estimate partners' trustworthiness. Each source represents a factor for trust establishment and can be used to specify a requirement within a policy. For example, in Figure 1, *Alice* can use four types of trust factors to evaluate *Bob*'s trustworthiness within the community: (i) she can compute *Bob*'s reputation based on the other members' past experiences with *Bob*, (ii) she can use recommendations of members she trusts, (iii) she can also ask confirmed members whether *Bob* is skilful in certain disciplines and what is his expertise level, and finally, (iv) she can ask *Bob* for credentials – he owns – to testify his identity, his skills or other properties. Then she can check the validity of these credentials from the *certification authority* that issued the credential (e.g. University). This example will be used throughout this article to motivate and illustrate different aspects of our approach.

2.2 Desiderata

The study of the above IC example has led us to identify important requirements for trust management, which are as follows.

Semantics. One of the characteristics of ICs is the great heterogeneity among its members. The problem lies in the fact that trust is built upon statements (e.g. credentials and reputations) that are manipulated by multiple parties, making semantic heterogeneity an important concern for trust management in virtual communities. The descriptions of trust factors must have a well-defined semantics [21], so that the policies specified based on this factors and the information used to satisfy this policies can be understood by all. For instance, *Bob* must have confidence that when he considers that the credentials he owns satisfy *Alice*'s policy, *Alice* should make the same conclusion.

Subjectivity. The heterogeneity among the community members raises also the problem of trust subjectivity. For instance, in Figure 1, Alice and Bob may rely on different trust factors when specifying the policy they use to protect a common resource. Historically, the majority of trust management models are built upon a controlled – and limited– set of trust factors [3, 13]. Such limitation represents a lock for subjectivity and constraints the expressiveness of trust policies. Our trust management system should allow the community members to specify policies using various trust factors, characterizing the subjective nature of trust.

Adaptivity. ICs are open environments that members can spontaneously join and leave and where resources are unpredictably created, updated and destroyed. In such settings, future interactions are impossible to predict, making the specification of trust policies hazardous and risky [13]. The question one should ask, in this context, is not *"how efficient the policy I specified is?"* but *"how adaptive, in response to an ever-evolving environment, it could be?"*. For example, the policy that Alice is using relies on reputation (and other trust factors as well). The more Alice is aware of collusions risk, the less she will give importance to reputation factor in her final decision. So, her policies need to be automatically adapted to such environment changes. Otherwise, she could make inappropriate decisions. Thus, policies within virtual communities can not be specified *ad vitam æternam* (specified once, executed for ever). The traditional *dogmatic* vision of trust policies is neither possible nor desirable and should even be proscribed.

Social-Awareness. Trust is closely tied to values, traditions, and norms of the society. Many sociologists [4, 17] demonstrated the impact of social influence on our practices and attitudes. Trust practices are not an exception but, to the best of our knowledge, no research initiative was conducted in that direction in terms of trust management. *"Why should an individual integrate collective requirements when making trust-decisions?"*. Principally for two reasons: (i) *social exclusion* and (ii) *competitive exclusion*. Both effects arise when the decision of a member is too far apart from the decision of the other members of his community. For instance, members of the community in Figure 1 co-regulate the access to resources they produced collaboratively. If Bob does not comply with the requirements of his community (i.e. collective policies), his behaviour will be considered as an antisocial deviant behaviour, and can expose Bob to *social exclusion*. Also, ICs represent competitive environments where different communities are competing in solving the same problem. In such situation, using a too restrictive or a too permissive policy exposes agents to *competitive exclusion*. Therefore, trust decision made by an agent in its community based on its individual requirements should be tempered by the integration of collective requirements.

3. FOUNDATIONS: SYSTEM OVERVIEW

In the context of our Innovation Community scenario, the system model (cf. Fig. 2) is represented by a multi-agent system $\mathcal{S}$. $\mathcal{S}$ is defined by a set $\mathcal{A}$ of agents a, a set $\mathcal{C}$ of communities c, a set $\mathcal{O}$ of operations o, a set $\mathcal{R}$ of resources r, a set $\mathcal{I}$ of interactions $\imath$, a set $\mathcal{F}$ of feedbacks e and an ontology $\mathcal{T}$ of trust factors in $\mathcal{S}$.

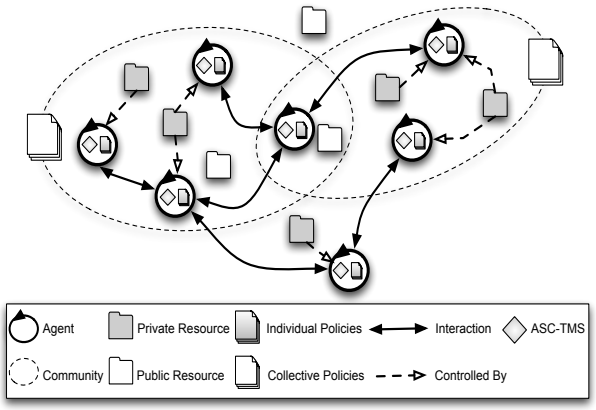

Figure 2: System model overview

Agents are autonomous entities assisting humans in their activities by performing operations on resources. No agent has direct control over other agent's operations and trust decisions. And each agent is endowed with a trust management system for its trust assessment tasks. $\forall a \in \mathcal{A}$, $a = \langle \varepsilon_a^{\mathcal{A}}, \alpha_a, \beta_a, \kappa_a, \omega_a, \psi_a, \mathcal{TP}_a \rangle$, where, $\varepsilon_a^{\mathcal{A}}$ is the agent's unique identifier in $\mathcal{S}$, α_a is a set of credentials attesting its various properties, $\beta_a \in 2^{\mathcal{C}}$ is the set of communities that a *belongs* to, $\kappa_a \in 2^{\mathcal{O}}$ is the set of operations that a can perform (i.e. *capabilities*), $\omega_a \in 2^{\mathcal{R}}$ is the set of resources that a *controls*, $\psi_a \in 2^{\mathcal{A}\times\mathcal{F}}$ is a set of feedbacks the agent a maintains about other agents' behavior after each interaction, and $\mathcal{TP}_a$ is a set of trust patterns (cf. Section 4 and 5 for more details about trust patterns). $\forall tp \in \mathcal{TP}_a$, tp includes individual policies ($p_a^i / p_a^i \in \pi_a$), associated meta-policies ($\mathcal{M}(p_a^i)$) and collective policies ($p_c^i / p_c^i \in \Pi_c$ and $c \in \beta_a$).

Agents from $\mathcal{S}$ can join together to form **communities** wherein they engage in frequent interactions in order to solve collaboratively a specific problem. When joining the community, each agent receives a set of *collective policies* representing common minimal conditions the community members' agreed on for the trust management. The *social compliance* of an agent depends on whether his decision is in accordance with this set of *collective policies* or not. Each community $c \in \mathcal{C}$ is represented by $\langle \varepsilon_c^{\mathcal{C}}, \Pi_c \rangle$ where $\varepsilon_c^{\mathcal{C}}$ is its unique identifier and Π_c is the set of collective policies.

Operations represent the actions an agent is able to perform on certain resources. Without loss of generality, we are considering only a limited set of operations (e.g. access, update, delete, grant and delegate).

Resources are artefacts that agents use during their activities by performing *operations* on them. $\forall r \in \mathcal{R}$, r is represented by $\langle \varepsilon_r^{\mathcal{R}}, \tau_r, \varsigma_r \rangle$, where $\varepsilon_r^{\mathcal{R}}$ is its unique identifier, $\tau_{\mathcal{R}}$ represents its type (e.g service, document, credential) and ς_r qualifies its sensitivity. A resource can be either *private* or *public*. *Public* resources can be manipulated by any agent within the community, while *private* resources' manipulation is limited to trusted agents.

Interactions can take place between an agent and the resources that comprise its environment, between an agent and an other agent and between and agent and its community. The first form of interaction is achieved by invoking operations on resources (e.g. access, update and delete), while the latters are achieved through message exchange. Each message $m \in \mathcal{I}$ is defined by $\langle a_s, a_r, \tau_m, \theta \rangle$, where a_s, a_r are respectively the sender and the receiver of the message, $\tau_m \in \{request, inform, reply\}$ is the message's performative type and θ its content. Given that trust decisions mainly concern *requests*, we will focus on such messages. *Requests* are interaction messages where the content θ is a pair $\langle o, r \rangle$ (called the request pattern) stating that a_s is asking a_r a permission to perform an operation $o \in \mathcal{O}$ on the resource $r \in \mathcal{R}$ that a_r controls.

A **permission** is a *reply* to a *request* message where the content θ is of the form *allow(o, r)* (resp. *deny(o, r)*) stating that the controller a_c accepted (resp. refused) the requester's request.

The **Trust Ontology** $\mathcal{T}$ represent a semantic description of trust factors available in $\mathcal{S}$. $\forall f_i \in \mathcal{T}$, f_i is a concept defined by $(\tau_{f_i}, \Delta_{f_i})$ where τ_{f_i} is the type of f_i and Δ_{f_i} its domain (i.e. the set of values f_i can take). Each trust factor describes agent's characterizing traits (e.g. identity or reputation) on which restrictions can be specified using policies.

4. ADAPTIVE AND SOCIALLY-ENABLED TRUST MODEL

Now that the system model has been introduced, we present how trust is built and managed within such system using the proposed trust model (cf. Fig. 3). This model is implemented in each agent of the system as the Adaptive and Socially-Compliant TMS (ASC-TMS).

4.1 The Model Overview

Each time an agent $a \in \mathcal{A}$ joins a community $c \in \mathcal{C}$, he is notified about the existence of collective policies he should comply with. Both collective policies (Π_c) and the agent's individual policies (π_a) are sorted with respect to the request pattern they handle, then stored into *trust patterns*. Once a receives a request $\langle b, a, request, \langle r, o \rangle \rangle$ from an agent b asking him the permission to perform an operation o on a resource r, he forwards it to its trust management system in order to handle it. The ASC-TMS identifies the request pattern and selects the appropriate *trust pattern*. First, the selected *individual policy* is adapted with respect to a's *Personal Preferences* and c's current *Business Context*, then this policy is combined with the related *collective Policy* in order to yield with the community trust requirements (cf. Section 5.3), and finally the generated policy is evaluated and a trust value is computed.

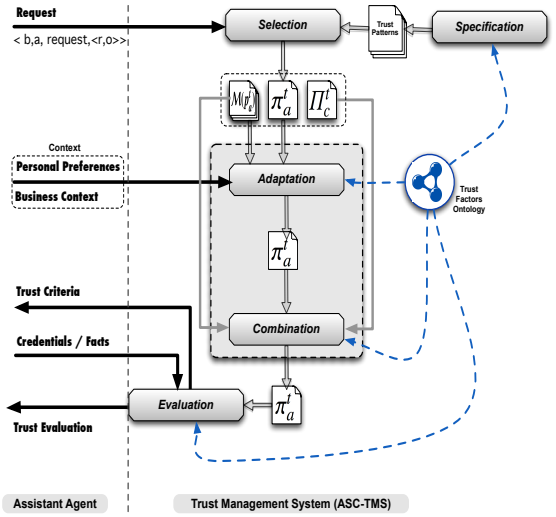

Figure 3: The Trust Management Model for the agent a within c

Policies are statements that specify, for each interaction pattern, on which conditions trust decisions are made. Each policy p represents set $\{tc_1, tc_1, \ldots, tc_n\}$ of Trust Criteria tc_i that constitute a conditions stated by the policy.

A **Trust Criterion** tc_i ($i \in [1, n]$) is a triplet $\langle \tau_{f_i}, \aleph_i, w_i \rangle$. τ_{f_i}, is the trust criterion type and corresponds to existing trust factors $f_i \in \mathcal{T}$, $\aleph_i$ is a threshold value among possible values ($\aleph_i \in \Delta_{\tau_{f_i}}$), while $w_i \in \mathbb{N}^*$ represents the importance (i.e. weight) of tc_i within the trust level computed when evaluating the policy.

Adaptation Meta-Policies ($\mathcal{M}(p_a^t)$) are event-condition-action (ECA) rules that agents use when adapting their individual policies p_a^t to yield their *personal preferences* and to fit the surrounding *business context*. Each rule is of the form:

$$Upon\langle event\rangle : If\langle condition\rangle \leftarrow Do\langle action\rangle \qquad (1)$$

The $\langle condition\rangle$ is a general expression, typically a conjunction or a disjunction of literals, to be checked before the execution of the rule. These conditions represent filters that are defined over the community *business context* (e.g competitive, cooperative, . . . , bloody-minded pairs), the agent *personal preferences* (e.g. tolerant, uncompromising, naive, . . . , wise agent), the $\langle action\rangle$ is one or more adaptation action specified by the agent in order to adapt the policy, while $\langle event\rangle$ is a triggering event.

Trust Patterns ($\mathcal{TP}$) have been introduced in our model to overcome limitations we faced when we tried to specify policies in ICs scenarios. The main challenge was that in traditional approaches, each resource is governed by a specific policy. But in virtual communities, resources are dynamic and unpredictable (e.g. spontaneously created and their sensitivity can change over time). This constraint obliged us to consider a new model where policies are specified independently from the resource they govern. So, the question was *how to protect resources that have not yet been created?* and our solution was to focus on requests types rather then the resources themselves.

Context information are domain-dependent beliefs agents rely on when adapting their policies. In our approach we used, but are not limited to, *personal preferences* and *business contexts*. **Personal Preferences** are personal input parameters an agent uses to know whether he has to adapt a policy or not. The **Business Context** has also a great influence on trust decisions. Whether the community is highly competitive or fairly cooperative, trust requirements can be formulated in different ways affecting the final decisions. This model enables agents to make context-aware trust decision based on such indicators. In this paper, for simplicity reasons, only few business parameters are considered. They are domain-dependent but most of them can be generalized to handle other VCs scenarios. Our *Business Context* is qualified by the number of agents requesting the resource, the number of agents providing it, the population of the community and the past experience with the other members of the community, which constitutes a good barometer of the relevance of indicators like *reputation* and *recommendation*.

4.2 The Trust Factors Ontology

The Trust Factors Ontology $\mathcal{T}$ (see Fig. 4) is used to capture and represent essential requirements that are the basis for establishing trust between agents within virtual communities. The first level root concept represents a generic trust factor type, it is further divided in two sub-types *proofs* and *indicators*: proofs represent certified information, while indicators include all possible facts stored internally or gathered from external sources. Even if they are impossible to check and may be subject to tempering, *indicators* are valuable hints for trust evaluation in many situations (e.g. when no certification authority exists).

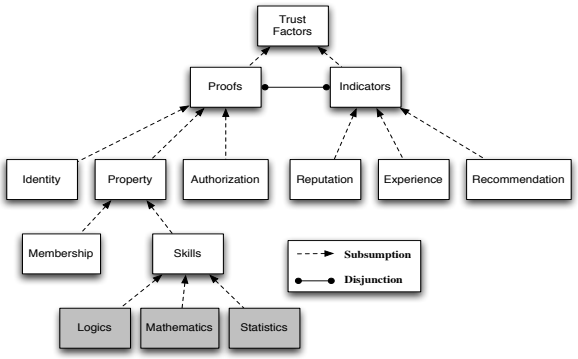

Figure 4: Fragment of the Trust Factors Ontology: white concepts are domain-independent while grey ones are domain-dependent

The ontology $\mathcal{T}$ is used in the model (cf. Fig. 3) at different levels. (i) For policies specification where τ_f are used as types and the $l \in \Delta\tau_f$ as thresholds values; (ii) for policies adaptation where $\mathcal{T}$ offers values ordering agents refers to in order to strengthen or weaken their policies; and (iii) while combining individual and collective policies where it offers referential values set for choosing the most restrictive policy. And finally, when evaluating policies where provided credentials are compared with respect to the required ones.

5. IMPLEMENTING ADAPTIVITY AND SOCIAL-COMPLIANCE

This Section provides an overview of the adaptive and socially-enabled aspects of our trust model. While policies are used to specify trust requirements (i.e. trust factors type, required values and weights of these factors), meta-policies encapsulate mechanisms to adapt and combine these policies in response to specific context parameters. We first briefly present how policies and meta-policies are specified then we describe how policies are adapted and combined.

5.1 Policies specification

We adopted logic programming and represented policies as structures. Prolog is especially interesting as it is declarative and can be used to express any criteria [6]. Each policy is policy structure is of the form:

$$policy([tc(\tau_{f_1}, \aleph_1, w_1), \ldots, tc(\tau_{f_n}, \aleph_n, w_n)])_{[Issuer, Pattern]}$$

Where $[tc(\tau_{f_1}, \aleph_1, w_1), \ldots, tc(\tau_{f_n}, \aleph_n, w_n)]$ represent a set of trust criteria, $Issuer$ is an agent idenfier ($\varepsilon_x^{\mathcal{A}}$) if the policy is an individual policy or a community identifier ($\varepsilon_x^{\mathcal{C}}$) if the policy is a collective one and $Pattern$ is the request pattern the policy handles.

5.2 Policies adaptation (Meta-policies)

As mentioned previously, meta-policies state how a policy could be adapted to handle a specific context. The meta-policies presented in this paper are specified by means of Jason (i.e. AgentSpeak) plans. Each plan is represented as follows.

$$triggering\_event : context \leftarrow body \quad (2)$$

The mapping between Jason plans and ECA rules is relatively trivial. The *triggering_event* denote the purpose for that plan (e.g adapting a particular policy) and is followed by a conjunction of belief literals qualifying a context (e.g. when the adaptation should be launched). The body of a plan is a sequence of basic actions the agent has to achieve when the plan is executed, if the context conditions hold. The key feature in using Jason plans lies in the possibility to execute legacy code through *internal actions*. In this work, we use internal actions to execute adaptation operations presented in our trust model (for more details on AgentSpeak plans and internal actions concepts see [7,18]). The result of the execution of each meta-policy affects the original policy as follows.

1. `.add`$(\langle \tau_{f_i}, \aleph_i, w_i \rangle)$: this action allows to add a trust criterion to the policy.
2. `.set`$(\langle \tau_{f_i}, \aleph_i, w_i \rangle)$: this action is used to update existing *trust criteria* values.
3. `.del`(τ_{f_i}): this action removes all *trust criteria* of the type τ_i from the policy.
4. `.lower`(τ_{f_i}) /`.increase`(τ_{f_i}): are actions used to, respectively, lower and increase the weight of a criteria within a policy.
5. `.restrict`(τ_{f_i}) / `.relax`(τ_{f_i}): are complementary actions. they make use of the *ontology* to find a lower (resp. higher) value for the specified *trust criterion*. If such value exists, a `.set()` with the new value is called otherwise the weight of the *trust criterion* is lowered (resp. increased). When the weight becomes null, the *trust criterion* is removed from the policy.
6. `.merge`(π_a, Π_c): This action combines the individual policy π_a with the collective policy Π_c where $a \in \mathcal{A}$ and $c \in \beta_a$. The resulting policy is at least as restrictive as the most restrictive policy (The algorithm used to implement this action is detailed in the the Section 5.3 below).
7. `.integrate`(π_a, Π_c): Similarly to the previous action , the combined policy is here at least as restrictive as the individual policy π_a.
8. `.incorporate`(π_a, Π_c): Complementary to the previous action , the combined policy is here at least as restrictive as the collective policy Π_c.

5.3 Policies combination (Algorithms)

The ASC-TMS ability to combine individual and collective policies represents the socially-enabled aspect of our trust model. The mapping we make between *social influence theory* and *trust management theory* concepts is concretized through the combination actions (6) to (8) presented above. For instance, we used the heuristic sketched below to implement the `.merge()` action in our proof-of-concept framework. The algorithm is an extension of the "Deny Override" combination algorithm used on the XACML policy language [16]. The combination algorithms we opted for are relatively simple as our focus in this paper was to present the overall abstract vision of our trust model.

Function Merge(π_a^t,Π_c^t)

$CP \leftarrow \Pi_c^t$; $IP \leftarrow \pi_a^t$; $P \leftarrow \emptyset$; $i \leftarrow 0$
while $CP \neq \emptyset$ **do**
  foreach $tc_x \in CP$ **do**
  if $\tau_{f_x} \notin IP$ **then**
  $tc_i \leftarrow tc_x$, $P \leftarrow tc_i$
  $CP \leftarrow CP - \{tc_x\}$
  $i \leftarrow i + 1$
while $IP \neq \emptyset$ **do**
  foreach $tc_y \in IP$ **do**
  if $\tau_{f_y} \notin CP$ **then**
  $tc_i \leftarrow tc_y$, $P \leftarrow tc_i$
  $CP \leftarrow CP - \{tc_y\}$
  $i \leftarrow i + 1$
forall the $tc_x \in CP$, $tc_y \in IP$ *s.t.* $\tau_{f_x} = \tau_{f_y}$ **do**
  $P \leftarrow tc(\tau_{f_x}, max(\aleph_x, \aleph_y), max(w_x, w_y))$
  $CP \leftarrow CP - \{tc_x\}$
  $IP \leftarrow IP - \{tc_y\}$
  $i \leftarrow i + 1$
return P

5.4 Policies evaluation

Let $p_a^t = policy([tc(\tau_{f_1}, \aleph_1, w_1), \ldots, tc(\tau_{f_n}, \aleph_n, w_n)])_{[a,t]}$ be the policy used by the agent $a \in \mathcal{A}$ to handle requests of type t. p_a^t evaluation represent a function $\mathcal{G}(p_a^t)$ that evaluates the trust level l relative to an instance request q of type t. Trust Level associated to r is computed by:

$$\mathcal{G}(p_a^t) = \frac{\sum_{i=1}^{m} \mathcal{G}(\tau_{f_i}, \aleph_i, w_i)}{\sum_{i=1}^{m} w_i} \quad (3)$$

Where $\mathcal{G}(\tau_{f_i}, \aleph_i, w_i) \in [0, w_i]$ is the weighted evaluation of the constraint $\aleph_i$ satisfaction on the criterion of type τ_i. Relative importance assigned to each trust criterion is modelled as the weight w_i of a criterion within the policy. Each criterion is then evaluated and returns one (1) or zero (0), respectively, whether the criterion is fulfilled or not. This result is then multiplied by the w_i and divided by the sum of all w_i in p_a^t.

5.5 An applicative scenario

Eurêka is an *Open Innovation* platform where each user is provided with an agent that assists him in his activities. Each agent is endowed with an ASC-TMS that allows him to make *adaptive* and *socially-compliant* trust decisions. *Eurêka* includes a wide variety of trust factors that users can rely on to evaluate each other trustworthiness. But for simplicity, we consider the situation where only identities, capabilities, recommendations and reputations are considered. In the following, we use capital letters to refer to human users (e.g. *Alice*) and lower case to refer to assistant agents (e.g. *alice*).

Alice, *Bob* and *Charlie*, are three users of the *Eurêka* platform to which was associated, respectively, three agents *alice*, *bob* and *charlie*. They joined together and created a community $\mathcal{K}$ where they engage to find a solution to one of the proposed problems (e.g *"new incentives to reduce energy consumption"*). To that aim, each member get involved in the production of specification documents representing the solutions they are working on. Both three agents can issue permissions that allow other agents to access, update and/or delete these documents, making the use of collective policies paramount for the community cohesion.

Let $q = \langle access, object \rangle$ be a request pattern used to handle requests about access operations on document resources. The collective policy the members of $\mathcal{K}$ must comply with when taking decisions regarding the requests pattern q is as follows:

$$policy([tc(identity, ultimate, 3), tc(reputation, 0.5, 1)]_{[\mathcal{K},q]}$$

This policy states that when the members of $\mathcal{K}$ grant access to common documents, the requester should have an ultimate identity credential (with respect to the PGP model [1]) and a reputation value of 0.5. The policy defines also the weight of each criterion among the decision. Similarly, the individual policy *alice* uses to handle requests of the same type can be stated as follows:

$$policy([tc(identity, complete, 1), tc(statistics, 0.5, 2),$$
$$tc(reputation, 0.7, 1), tc(recommendation, 0.2, 2)])_{[alice,q]}$$

Here, *alice*'s policy is more restrictive as it requires two additional conditions namely, statistics capabilities and recommendations. Let $\langle dave, alice, request, \langle access, energy \rangle \rangle$ be a request from *dave* asking *alice* a permission to access the *energy.txt* private document. The request here is of type q but the policy *alice* specified is too generic and should be adapted to the specific context in which she received the request. To that aim, *alice* can use meta-policies like the one given below to adapt her policies.

$$+!adapt(dave, "energy.txt", q) :$$
$$?community(Com) \wedge .count(member(\_, Com), C) \wedge$$
$$threshold(minMembers, T) \wedge (C < T)$$
$$\leftarrow ?policy(IP)_{[alice,q]}, .relax(reputation),$$
$$.del(recommendation).$$

With the previous meta-policy, *alice* says to her ASC-TMS to lower the values required for the reputation (to 0.6) and to remove the recommendation trust criterion when the community population is below a certain threshold. In the same way, she makes use of meta-policies to specify how compliant she wants to be toward the community she belongs to. Such meta-policy could be defined as follows:

$$+!combine(q) :$$
$$?myCommunity(Com) \wedge attractiveness(Com, A) \wedge$$
$$thresholdOf(attractiveness, T) \wedge (A > T)$$
$$\leftarrow ?policy(IP)_{[alice,q]}, policy(CP)_{[Com,q]},$$
$$.integrate(IP, CP).$$

alice is using the above meta-policy to decide whether to comply with the collective policy based on the attractiveness of the community. So if the context fits, the adapted policy she will get after applying the above *adaptation* and *combination* meta-policies should be as follows:

$$policy([tc(identity, ultimate, 3), tc(statistics, 0.5, 2),$$
$$tc(reputation, 0.6, 1)])_{[alice,q]}$$

In the above example, alice exhibited a compliant behavior as the policy she used integrates all the requirements imposed by her community. The `.integrate()` action ensures such behavior. But alice had also the possibility to adopt different behaviors by using other combination actions. For instance, the `.incorporate()` action gives the priority to individual policies to make sure the compliance of an agent with his personal requirements at the risk of deviance toward his community. Further, the `.merge()` action allows agents to exhibit a moderate behavior by complying with both individual and collective policies. The social-compliance concepts drawn from the *social influence theory* [4, 17] we presented here was intentionally simplified for the sake of clarity. We refer the reader to [22, 23] for more details on these aspects, the algorithms and heuristics we used to adapt policies, how the resulting policies are evaluated and how trust decisions are made based on these evaluations.

6. CONCLUSION

The last decade has seen an increasing interest on trust that gave rise to numerous researches and many definitions (cf. [3]). Existing trust models can be categorized into two distinct families: *Hard Trust Models* where trust is established using credentials and trust policies (e.g. [5] and [12]) and *Soft Trust Models* where trust is built upon pairs experiences, recommendations and reputation (e.g. [9], [20], [13] and [14]. Full integration of soft and hard trust models has not been done yet. However, [6] and [15] paved the way recently for such perspective by developing new *Hybrid Trust Models*. This work is an additional step toward that direction as it proposes an ontology-based *hybrid trust model* that enables agents to grasp the meaning of trust factors.

In this paper, we introduced the concept of social compliance for trust management within virtual communities. We outlined a new trust model where individual requirements are adapted and merged with collective ones. The originality of this proposition lies in the joint use of flexible trust policies and adaptation meta-policies to capture and adapt trust requirement. The concept of adaptive policies has already been investigated by many scholars (e.g. [19]). However, in many cases, the proposed solutions are built upon a known and finite universe of resources and entities which is a hard constraint if we consider open environments like virtual communities. Further, the idea of combining different policies has also been studied (e.g. [11]). However, the focus of existing approaches on policies integration was principally on static security policies. The combination issues addressed in this paper could not be managed by former approaches as the policies to be combined are specified, adapted and managed.

The next steps in our work would be to improve the "quite simplistic" heuristics used for policies adaption and combination by drawing some mechanisms from the prolific literature on trust and security policies (e.g. policies integration and combination algorithms). We also would like to extend this model in order to integrate other effects of the social influence theory [4]. Another interesting issue is to investigate how individual trust requirements (policies) could be integrated into collective ones. The idea would be to study how agents may influence / trigger adaptation or evolution of collective policies, and which mechanisms can they use to build collective policies from individual ones. Such feature is particularly interesting for decentralized and self-organized communities like the dynamic communities of experts that are regularly created in the innovation community's scenario studied in this article.

7. REFERENCES

[1] A. Abdul-Rahman. The PGP Trust Model. *The Journal of Electronic Commerce*, 1997.

[2] M. Antikainen and H. Väätäjän. Collaboration in open innovation communities. In *Proceedings of the XXII ISPIM Conference*, 2008.

[3] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 2007.

[4] S. E. Asch. Opinions and social pressure. *Scientific American*, 1955.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, 1996.

[6] P. Bonatti, C. Duma, D. Olmedilla, and N. Shahmehri. An integration of reputation-based and policy-based trust management. In *In Semantic Web Policy Workshop*, 2005.

[7] R. H. Bordini and J. F. Hübner. Semantics for the jason variant of agentspeak (plan failure and some internal actions). In *Proceeding of the 19th European Conference on Artificial Intelligence*, 2010.

[8] T. Dimitrakos, B. Matthews, and J. Bicarregui. Towards security and trust management policies on the web, 2001.

[9] R. Falcone and C. Castelfranchi. Social trust: a cognitive approach. *Trust and deception in virtual societies*, 2001.

[10] E. GRACIA and J. HERRERO. Determinants of social integration in the community: An exploratory analysis of personal, interpersonal and situational variables. *Journal of Community Applied Social Psychology*, 2004.

[11] D. D. He and J. Yang. Security policy specification and integration in business collaboration. In *IEEE International Conference on Services Computing (SCC 2007)*, 2007.

[12] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. Access control meets public key infrastructure. In *IEEE Symposium on Security and Privacy*, 2000.

[13] A. Jøsang. Trust and reputation systems, 2007.

[14] K. Krukow, M. Nielsen, and V. Sassone. A framework for concrete reputation-systems with applications to history-based access control. In *Proceedings of the 12th ACM conference on Computer and communications security*, 2005.

[15] A. J. Lee and T. Yu. Towards a dynamic and composable model of trust. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, SACMAT '09, 2009.

[16] P. Mazzoleni, E. Bertino, B. Crispo, and S. Sivasubramanian. Xacml policy integration algorithms: not to be confused with xacml policy combination algorithms! In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, SACMAT '06, 2006.

[17] S. Moscovici, G. Mugny, and E. V. Avermaet. Perspectives on minority influence. *European studies in social psychology*, 1985.

[18] A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language, 1996.

[19] T. Ryutov, L. Zhou, C. Neuman, N. Foukia, T. Leithead, and K. E. Seamons. Adaptive trust negotiation and access control for grids. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[20] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents*, 2001.

[21] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *In 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.

[22] R. Yaich, O. Boissier, P. Jaillon, and G. Picard. Social-compliance in trust management within virtual communities. In *3rd International Workshop on Web Intelligence and Communities at the International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2011)*, 2011.

[23] R. Yaich, O. Boissier, P. Jaillon, and G. Picard. Trust Management within Virtual Communities: Adaptive and Socially-Compliant Trust Model. Technical report, FAYOL - ENSM-SE, 2011.