Impact of Social Influence on Trust Management within Communities of Agents

Reda Yaich[1], Olivier Boissier[2], Gauthier Picard[2], and Philippe Jaillon[3]

[1] IMT Atlantique, Lab-STICC, UMR CNRS 6285, Institut Mines-Télécom
reda.yaich@imt-atlantique.fr
[2] Mines Saint-Étienne, Laboratoire Hubert Curien UMR CNRS 5516, Institut Henri Fayol
{boissier,picard}@emse.fr
[3] Mines Saint-Étienne, Institut Henri Fayol
jaillon@emse.fr

Abstract. In the real world as in the virtual one, trust is a fundamental concept. Without it, humans can neither act nor interact. So unsurprisingly, this concept received in the last years a growing interest from researchers in security and distributed artificial intelligence that gave rise to numerous models. The principal aim of these models was to assist users in making safe decisions at the individual level. However, studies have shown that the behavior of an individual within collective structures (e.g., a group, a community, a coalition or an organization) is affected (directly or indirectly) by the behavior of other members, creating a social influence dynamics within these structures. In this article, we study the impact of social influence phenomena when they are applied to trust management within open distributed communities of self-organized and self-governed agents.

1 Introduction

Virtual communities (VCs) are socio-technical systems wherein groups of distributed individuals (human and/or artificial) cooperate in achieving common objectives and goals [39, 52]. The achievement of these goals requires the collaboration of all members through resources, skills and/or knowledge sharing. A collaboration always bears the risk that one partner exhibits non cooperative or malicious behavior, making trust a central concern.

Trust and *Trust Management* have been extensively investigated in the last fifteen years by researchers from Security and Distributed Artificial Intelligence [5, 48, 52]. By making reference to the concept of trust, research on trust management has made explicit the parallel between human societies and distributed computer systems. So, many of what we can learn about human communities may be applicable in virtual ones. However, little work has been done so far with respect to *what mechanisms humans use to deal with situations in which the trust decision they make may impact other members of the community*. According to Pearlman, "a key problem associated with the formation and operation of distributed virtual communities is that of how to specify and enforce community policies" [39]. Collective policies are core elements of a virtual community. They are used by the members of the community to govern the access to communities and shared resources. However, in the current literature, solely individual requirements (i.e., individual policies) are used and considered during the decision process. Nevertheless, when it comes to social structures with an explicit collective dimension, such approaches must be prescribed [1, 28, 52]. In fact, the consequences of decisions made by a member can affect the other members. Making a collective management of trust a necessity. Indeed, several studies conducted in sociology evidenced the existence of *social influence* phenomena making individuals complying with collective norms and policies, even at the expense of their own interests [46, 1].

The work we report in this article shows how such social influence phenomena can be used to design a socially-aware trust management for self-organized and self-governed communities of agents. The objective is to evaluate the impact of a trust management system, that is safe at

the individual and the collective level as well, on the dynamics of these communities.

More concretely, the question we're trying to answer in this work is *whether allowing agents of a community to make evolve collective policies, they are subject to, could produce beneficial effects on communities in terms of number and size.*

To that aim, in Section 2, we first review the related works on trust and social influence. In Section 3, we present the multi-agent based virtual community that underpins our trust management model, and expound how our Adaptive and Socially-Compliant Trust Management System (ASC-TMS) assists Virtual Communities' members in building, enforcing and evolving collective policies. Finally, in order to demonstrate the benefit of our approach, we simulated our model and empirically studied its impact on the dynamics of these communities, in Section 4. The obtained results show a positive effect, foreseeing a beneficial impact on the performance and stability of theses communities. We conclude this article in Section 5 with a discussion and some research perspectives.

2 Literature Review on Trust Management

Research on trust has a long tradition in many disciplines such as philosophy (e.g., [19, 40, 27]), Psychology (e.g., [14, 41]), Sociology (e.g., [29, 30, 35]), Economy (e.g., [50, 34, 33]). In Computer Sciences, the necessity of trust appeared with the advent of distributed computing (e.g., Internet) and the increasing use of electronic interpersonal interactions rather than face-to-face ones, especially when interacting partners are software agents [16]. In such settings, the inability to obtain complete information, that is inherent to any distributed system, has driven research to think beyond security. However, in computer science, it is not sufficient to just define/understand trust (what other disciplines mainly focused on). For automation, the concept of trust must be *formally* conceptualized and represented. The word *formal* here means that this concept should be captured, represented and manipulated by software programs. As a consequence, in the last decades, research in trust management witnessed an increasing interest in formalization, implementation and evaluation of theories introduced by scholars from the disciplines aforementioned.

Trust management[4] has been mainly investigated in *distributed artificial intelligence* (DAI) and *security* disciplines wherein uncertainty constitutes the most challenging issue. In what follows, we provide a overview of what have been proposed in both disciplines. This review aims also at, introducing basic concepts we will rely on in this article, and motivating our contribution in the discussion section (i.e., Section 2.3).

2.1 Trust in Distributed Artificial Intelligence

Trust models in DAI fall into three categories; *probabilistic models*, *reputation models* and *socio-cognitive* models.

Marsh's work [32] on trust represents the first comprehensive and formal model of trust using a **probabilistic approach**. Marsh experimented in his doctoral thesis the use of trust as a basis for cooperation among autonomous agents. Starting from the assumption that "neither full trust or distrust are actually possible" in such systems, he proposed a complex calculation algorithm that computes a continuous value of trust based on several factors. This value represents the probability that A will behave "as if" he trusts B. Manchala developed the first trust model that

[4] Term initially coined by Blaze et al. in late 90's.

explicitly uses the notion of risk [31]. Manchala specified a decision matrix in which he used the cost of the transaction, the expected outcome, and the history of the transactions. These factors are then used together in a probabilistic function to determine whether a transaction with a partner should be conducted or not.

Reputation is the social evaluation of a group, a community or a society of agents towards the trustworthiness of an individual [44]. In DAI, and more particularly in multi-agent systems, reputation has been considered as a substantial dimension of trust [24]. In *ReGreT* [44], Sabater and Sierra used three factors based on which trust was computed: *the direct experience*, *the global reputation* and an ontological *fine-grained reputation* which defines reputation values for each trait of the individual using the ontology. In FIRE [23], the authors compute trust based on past experiences, the role of the agent, its reputation and a kind of certified reputation. Vercouter and Muller proposed the LIAR model to process recommendations among multi-agent systems [48, 49].

The model proposed by Castelfranchi and Falcone is the prime trust model that explicitly stressed the importance of the **socio-cognitive dimension** of trust [15]. They defined trust as a mental state based on which artificial agents can make delegation decisions within multi-agent systems. The authors consider trust as a combination of beliefs and goals that constitute this mental state.

2.2 Trust in Security

In security, several access and/or usage control models (e.g., [45, 26]), have been successfully used to prevent unauthorized access or manipulation of sensitive, private or confidential resources. However, most of these models operate under a known and finite universe of discourse. The system administrator needs to know beforehand what are the resources to be protected and what are the criteria (e.g., Identity, Role, Label) based on which users should be granted access to these resources. With the advent of Internet and the proliferation of open and decentralized systems, research community was obliged to think beyond security perimeters to enable interactions and collaboration under uncertain settings. Since then, several systems (called Trust Management Systems) have been proposed. We classify these systems into two categories: *Decentralised Trust Management (DTM)* and *Automated Trust Negotiation (ATN)*.

Blaze and Lacy were pioneers in using the concept of *trust management* in the security domain [5, 6]. They proposed the first **Decentralized Trust Management System**. The authors justified the use of trust in reference to the delegation mechanisms they used to address distribution in modern systems (e.g. Internet). Credentials (digitally signed documents) are used to allow an individual to express the trust relationship it has with another individual with respect to a particular issue (e.g. accessing a resource). For instance an individual A may issue a credential stating that he trusts another individual B to access a resource R he owns. Subsequently, B can express the trust he puts in another individual C with respect to the same issue. Now if C requests to A the access to R, he will provide the credential provided by B and the objective of the model is to evaluate whether there is a valid trust (delegation) chain from A to C about accessing the resource R. Analogously, several systems have been proposed in the last decades such as SULTAN (Simple Universal Logic-Oriented Trust Analysis Notation) [17] or Kagal and colleagues's [25] trust model for distributed security in the context of multi-agent systems.

Credentials which are implemented in DTM as identity or delegation certificates are used in trust negotiation systems to convey the identity and the attributes of the holder. So releasing a credentials imply the disclosure of such sensitive information. To that aim, Winslett and

colleagues [53] introduced the concept of **trust negotiation**. Trust is established through the gradual, iterative, and mutual disclosure of credentials and access control policies. This model has been proposed to leverage *privacy* issues that may arise when the disclosed credentials are sensitive. The authors build on exiting *decentralised access control models* to allow bilateral establishment of trust.

Trust-$\mathcal{X}$ is a trust management system that was designed for trust negotiation in peer-to-peer systems [2, 3]. The Trust-$\mathcal{X}$ engine provides a mechanism for negotiation management. The main strategy used in Trust-$\mathcal{X}$ consists in releasing policies to minimise the disclosure of credentials. So only credentials that are necessary for the success of a negotiation are effectively disclosed [47]. Trust-$\mathcal{X}$ provides also a mechanism to protect sensitive policies. Another novel aspect of this system consists in the use of trust tickets. Trust tickets are issued upon successful completion of a negotiation. These tickets can later be used in subsequent negotiations to speed up the process in case the negotiation concerns the same resource.

Bonatti et al. proposed a flexible and expressive negotiation model called *PROTUNE* (*PRovisional TrUst NEgotiation*) [9, 10]. This framework is a system that provides distributed trust management and negotiation features to web services. *PROTUNE* consists in a policy language (based on PSPL [8]) and an inference engine based on (PeerTrust [37]). The most innovative aspect of PROTUNE relies in its high degree of expressiveness. One of the main advances made by PROTUNE lies in the use of *declaration* along with credentials during the policy evaluation process. *Declarations* are the unsigned equivalent of credentials. They can also be considered as statements that are not signed by a certification authority. However, the most novel part of the project remains the policy specification language with combines access control and provisional-style business rules.

2.3 Summary and Discussion

In DAI, trust is *calculated*. It takes often the form of a *reputation* representing the evaluation of the reliability (in term of trustworthiness) of an individual performed by a group of individuals [48]. With respect to that, several models have been proposed (e.g., [44, 48, 23]). The common aim of these models was to improve the evaluation, propagation and the update of reputation measures.

In security, trust is *verified*. The mechanisms used for this verification have their roots in *access control* paradigm. Policies are used to express delegation of rights and/or privileges (i.e., trust relations) [5]. They are also used to express conditions that should be met prior to the establishment of any trust relationship [16]. The verification of trust uses certificates (i.e., *credentials*) provided by the interlocutor against active policies. If the information conveyed by these certificates satisfy the conditions stated in the policies, trust is established. Furthermore, in some situations, certificates may contain sensitive information. Since any disclosure of this information may be harmful for the owner, a *negotiation phase* is usually required prior to their release [51].

In security as well in DAI, the objective of state-of-the-art solutions is essentially to make the users' decisions as safe as possible at the personal and individual level. However, with the ever growing success of social networks and virtual communities, the ability of trust models to take into consideration the social dimension of these new environments is challenging how trust was traditionally computed or verified. Furthermore, several studies in Sociology demonstrated the existence of influence mechanisms, not only from the community to the individual (e.g., [46, 1]), but from the individual to the community as well (e.g., [36, 28]). Researchers such as Sherif [46], Asch [1], Moscovici [36] or more recently Latané [28] have identified two kinds of *influence*: *influence of the group on an individual*, and the *influence of an individual on the group*. The

first is called *majority influence* and the second *minority influence*. Majority influence has been defined as the attempt by the majority of individuals in a group (or a figure representing them [4]) to impose its common position on an individual or a minority of dissenters [46, 1]. Analogously, minority influence takes place when a minority exerts an influence that makes the group change its position in favor of the minority [36]. Nemeth explained that when majorities are faced with a consistent attitude (the one of the minority) they are confused and try to understand why the minority is so determined to express controversial opinions [38]. Minority influence is built upon this questioning that disconcerts the majority and leads in some situation to conversion.

The work presented in this article aims at reproducing theses phenomena and study their effect in the context of trust management within communities of Agents. We more specifically tried to reproduce the mechanisms described in *social influence theory* [28] to specify in which conditions collective policies emerge, are enforced and/or adapted.

3 Trust Management within Communities of Agents

The adoption of agent and multi-agent technologies is a trending approach for modeling and implementing collaborative, self-organized and self-adaptive virtual communities (VC) as well as social networks (SN) in general [11, 42, 18, 52]. In this section, we will describe the multi-agent-based framework that underpins our trust management model. This model has already been presented in [52] but the contributions introduced in this article builds on the top of that model, which requires the reader to have in mind basic understandings of our previous works (that we summarize here after).

3.1 Specification of Agents Communities

As illustrated in Figure 1, agents are used as first class abstraction of virtual communities participants.

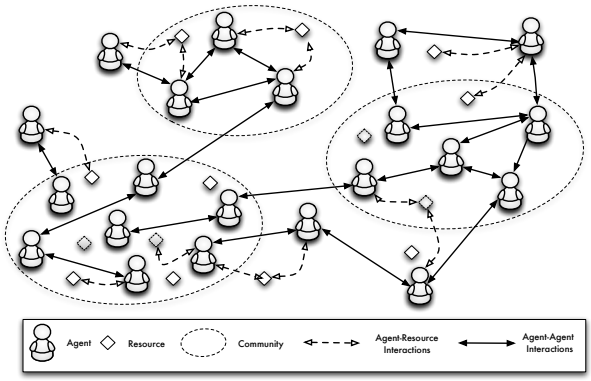

Fig. 1. An illustrative example of a population of agents with three multi-agent communities.

This multi-agent system $\mathcal{S}$ can be defined at a time t by a 5-uplet:

$$\mathcal{S}^t = \langle \mathcal{A}, \mathcal{R}, \mathcal{C}, \mathcal{I}, \Delta \rangle^t \tag{1}$$

where:

- $\mathcal{A}$ is the set of agents participating to $\mathcal{S}$,
- $\mathcal{R}$ is the set of resources that constitute the environment shared by the agents,

– $\mathcal{C}$ is the set of communities organizing and structuring the activities of the agents,
– $\mathcal{I}$ is the set of interactions between agents,
– Δ is the domain ontology.

In order to reduce the complexity of our descriptions, we will fully refrain from delving into MAS detailed properties of $\mathcal{S}$. Instead, we focus on the *agent* ($\mathcal{A}$), the *environment* ($\mathcal{R}$), the organization ($\mathcal{C}$) and the *interaction* ($\mathcal{I}$) elementary dimensions with respect to the VOWELS paradigm [13].

Agents represent autonomous (software or real) entities able to perform some tasks in line with their goals. $\mathcal{A}$ represents an heterogeneous population of agents. The heterogeneity of these agents is characterized in terms of the *properties* they possess. Properties represent the prevailing traits of an agent within the system.

The system *environment* is defined as a dynamic set $\mathcal{R}$ of passive entities called resources. $\mathcal{R}$ is dynamic because the resources are unpredictably created, updated and destroyed. The *content* that resources encapsulate represents data, services or any functionality that supports agents in their individual and collective activities.

In this system, we are particularly interested by the situations where, for some specific reasons, agents from $\mathcal{A}$ form several temporary ad-hoc groups. We denote the set of all possible groups in $\mathcal{A}$ by $\mathcal{C}$ where $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_m\}$. We refer to each member c_i of $\mathcal{C}$ as a *virtual community*. So virtual communities are sporadic and ephemeral organizations wherein agents are grouped to achieve a common objective. Our specification of communities is based on the $\mathcal{M}oise$ [21] meta-model. However, due to the paper format, we will not provide the detailed $\mathcal{M}oise$ specification of communities.

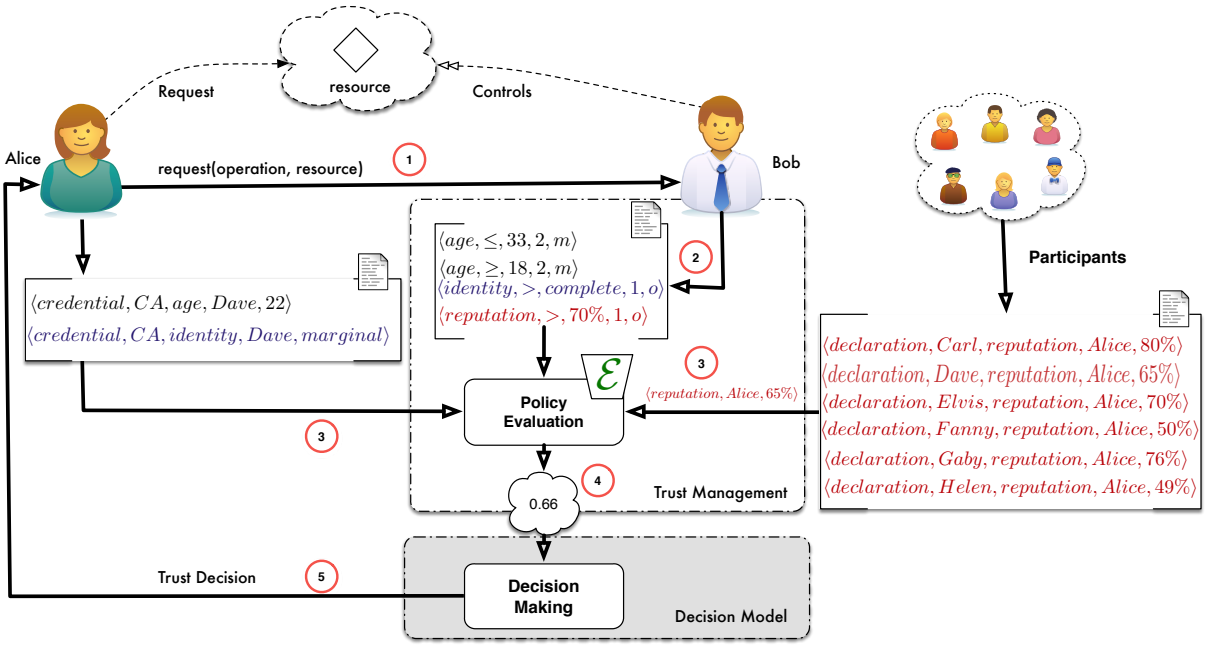

Fig. 2. Evaluation of a trust policy

In $\mathcal{S}$, each agent has a set G of (personal and/or collective) goals to which the agent has to some extent committed to. The achievement of each goal involves the performance of a sequence of actions (i.e., plans) on resources. These resources may be personal (i.e., owned by the agent) or non personal (i.e., owned by other agents or communities). So, trust becomes an issue in $\mathcal{S}$

since the fulfillment of agent's goals may require the access to sensitive resources, owned and governed by other agents. The trust issue arises also when the agents have to make decisions about admitting or not new members within their communities. To this aim, we present in the following section the trust management model we proposed to assist agents in their personal decision making [52]. Then we will show how we extended this model in order to take into consideration the social phenomena that we discussed in the introduction of this article: *how to enable a collective and socially aware management of trust decisions*.

3.2 Individual Management of Trust

In our model, (trust) policies are used to encapsulate the conditions (or constraints) that an agent (at the individual level) considers as necessary to establish trust with others. We represent policies as a collection of conditions, called *trust criteria*. A policy π is defined :

$$\pi_{Issuer}^{Pattern} = \{tc_1, tc_2, \ldots, tc_n\} \tag{2}$$

where $Issuer$ is the individual or the community that issued the policy, $Pattern$ is the Trust Pattern to which the policy applies, and $\{tc_1, \ldots, tc_n\}$ is a set of trust criteria. $TrustPatterns$ (or $Patterns$, for short) are used to specify the issue of the decision a policy can be used for. It is defined as a pair $\langle action, target \rangle$ that specifies which $action$ the requester wants to perform, and the artifact the action will affect. Trust criteria, denoted tc, are the building blocks of policies. They are atomic conditions[5] used to express threshold values over *trust factors*. A trust criterion tc is defined by:

$$tc = \langle f, op, v, w, t \rangle \tag{3}$$

where:

- f is a trust factor ($f \in \Delta$) that defines the properties of interlocutor agent on which we want to express constraints (conditions)
- $op \in \{=, >, <, \leq, \geq, \neq\}$ is a comparison operator,
- v is the criterion threshold value.
- $w \in \mathbb{Z}$ is the criterion weight,
- $t \in \{m, o\}$ is the criterion type; "m" means that the trust criterion is mandatory, while "o" means that it is optional.

To illustrate the trust policy language that we defined above, let's take the following sample policy:

$$\begin{aligned} \pi_{Bob}^{\langle join, community \rangle} = \{ & \langle age, \leq, 33, 2, m \rangle, \\ & \langle age, \geq, 18, 2, m \rangle, \\ & \langle identity, >, complete, 1, o \rangle, \\ & \langle reputation, >, 70\%, 1, o \rangle \} \end{aligned}$$

With this policy, Bob (the $Issuer$) governs his decisions about who can be admitted in the communities Bob belongs to. In this policy, four trust criteria are used to specify conditions over the age, the identity and the reputation of the candidate. For each criterion, Bob specified the threshold values, the weights and if these criteria are mandatory or not.Figure 2 illustrates how this policy is evaluated by Bob. First (1), Alice sends a request to Bob, asking him the

[5] By *atomic*we mean that these constraints cannot be further divided into other smaller conditions.

permission to perform an operation on a resource his owns/controls. Bob makes use of this Trust Management System (ASC-TMS) to (2) select the appropriate policy (based on the Pattern, which is a pair <Operation,ResourceType>). The evaluation of this policy is performed by the function $\mathcal{E}$. This function receives as inputs the policy and the required information to evaluate it. Here, two kinds of information are necessary; *credentials* and *declarations*. *Credentials* are certified documents stating a property of their owner. In this example, Alice need to provide Bob with credentials to prove her identity and her age (3). In the meanwhile, Bob's ASC-TMS collects information from other participants to compute the reputation value it requires to evaluate the policy (3). The evaluation function $\mathcal{E}$ (4) uses a weighted sum to compute the policy satisfaction degree based on the information collected by the model [52].

Concretely, let π_a^x be the policy used by the individual a to handle requests of the pattern x (e.g. $x = \langle read, text \rangle$). Now let us consider the request r initiated by b. After receiving the request, the controller a collects the trust information required by its policy and builds the requester's profile $a.\psi^b$.

The evaluation of π_a^x is achieved by applying the function $\mathcal{E}(\pi_a^x, \psi^b)$ as sketched by the formula 4 below, knowing that:

$$\pi_a^x = \{\langle f_1, op_1, v_1, w_1, t_1 \rangle, \ldots, \langle f_n, op_n, v_n, w_n, t_n \rangle\}$$

is the policy under evaluation, and

$$\psi^b = \langle q, b, \{\langle f_1, v_1' \rangle, \ldots, \langle f_m, v_m' \rangle\}\rangle$$

is the profile that a built about b. The policy evaluation is then performed as follows:

$$\begin{cases} \frac{\sum_{i=1,j=1}^{n} E(\langle f_i, op_i, v_i, w_i, t_i \rangle, \langle f_j, v_j' \rangle)}{\sum_{i=1}^{n} w_i} \\ \text{where } \langle f_i, op_i, v_i, w_i, t_i \rangle \in \pi_a^x, \langle f_j, v_j' \rangle \in \psi^b \text{ and } f_i = f_j \\ 0 \text{ if } \exists\ tc_i \in \pi_a^x / \ E(\langle f_i, op_i, v_i, w_i, t_i \rangle, \langle f_j, v_j' \rangle) = 0 \\ \wedge t_i = m \end{cases} \tag{4}$$

The function $E(\langle f_i, op_i, v_i, w_i, t_i \rangle, \langle f_i, v_i' \rangle)$ valued in $[0, w_{f_i}]$ performs the evaluation of a trust criterion tc_i with respect to a trust information ti_j. The correspondence between the trust criterion and the trust information is done based on the common trust factor they use. For instance, a trust criterion that constraints the age of the interlocutor is evaluated with the trust information that testifies such trust factor. So for each trust criterion, the function $\mathcal{E}$ looks for the corresponding trust information and evaluates whether this information satisfies the criterion or not. If it satisfies it, the computed value is 1 which is further multiplied by the weight of the criterion w_i, otherwise the computed value is 0. A trust criterion is computed to zero either because the trust information failed satisfying the criterion, or because the appropriate trust information has not been acquired. Further, the results of the evaluation of each trust criteria are summed and divided by the sum of all the weights. So the result represents the mean of the weighted sum and constitutes the satisfaction level of the controller policy with respect to the requester profile. Finally, if one of the criteria that are not satisfied is mandatory, the evaluation of the policy fails and returns 0.

In the above example, the information collected by Bob about $Alice$ (e.g., certificates, reputation values declared by other members) satisfy only 66% of the criterion expressed in his policy. This value is then forwarded to a separate trust decision model that will derive a trust decision to admit $Alice$ in the community or not based on it (5). This model is out of the scope of this paper (see [52] for more details) but we assume that Bob will use a kind of acceptance threshold value to allow automatic decision making.

3.3 Collective Management of Trust

In the previous example, our trust management system follows a classical scheme that allows Bob to take autonomous and sovereign trust decisions. However, within social structures such as multi-agent communities, such behavior could be risky for the community and consequently for each member as discussed in Section 1.

This is why we have extended our model with mechanisms to "*adapt policies to social context*". From the analysis of social theories, we identified *top-down pressure* by means of which the community forces its members to adapt their individual policies with respect to a collective one, but also a *bottom-up pressure* in which the individual policies of each member of the community shape the collective policy. This bottom-up pressure is then further detailed into two sub-pressures in reference to the process by means of which agents build the collective policy, and those they use to change it (i.e., evolve).

To adapt individual policies with respect to the social context (both bottom-up and top-down pressures), we define adaptation *meta-policies*. When applicable within the agents' communities, they are used to allow agents to trigger changes on their policies. Each meta-policy $m \in \mathcal{M}$ is defined as a function:

$$m : \mathcal{I}_a \times \Pi \times \mathcal{S}^t \times \mathcal{K} \rightarrow \Lambda^{\Pi} \tag{5}$$

A meta-policy m receives as input an instance of interaction $\mathcal{I}_a$ (involving a trust decision), a policy, an image of the system $\mathcal{S}$ at a time t and a condition expression to the appropriate set of actions to be performed on the policy.

Event-Condition-Action (ECA) rules [43] are used to implement adaptation meta-policies. Each meta-policy is composed of three parts: *the event*, *the conditions* and *the actions* as illustrated hereafter:

$$\langle event \rangle : \langle condition \rangle \leftarrow \langle action \rangle$$

The intuitive reading of an ECA meta-policy is *"if the event occurs in a context where the condition holds then the action must be executed"*. The event is generated by the trust management system (i.e., ASC-TMS) whenever it identifies one of the situations requiring policy adaptation (i.e. social pressures). The $\langle condition \rangle$ is a possibly empty set of conditions $k_i \in \mathcal{K}$. The conditions represent filters over the agent's *context* $\mathcal{S}^t$. The $\langle action \rangle$ is a sequence of adaptation operations through which the agent can change its policy. Finally, the $\langle event \rangle$ is the triggering event that will launch the adaptation. The triggering event allows to associate a name to each adaptation meta-policy.

In order to make policies adaptation possible, agents are endowed with a set of adaptation operations that can be applied to policies. Each operation is an action by means of which an agent can change the policy in a particular way. We identify two types of adaptation operations: ***simple operations*** and ***complex operations***. Simple operations concern the addition or deletion of trust criterion present in the policy, the change of the threshold values and/or the weights of existing criterion. Simple operations are defined as follows:

$$\forall \omega_i \in \Lambda^{\Pi},\ \omega_i : \Pi \rightarrow \Pi \tag{6}$$

In order to illustrate how meta-policies and simple operations are defined and used, let us consider the following example:

$$\begin{aligned} &Instantiate(\Pi_{Bob}, \_, \_, R, \_, \_) : R.value^t > R.value^{t-1} \\ &\leftarrow RestrictCriterion(\Pi_{Bob}, reputation) \end{aligned} \tag{7}$$

In this example, Bob uses a meta-policy that systematically restricts the reputation of the selected policy each time the requester resource value increases.

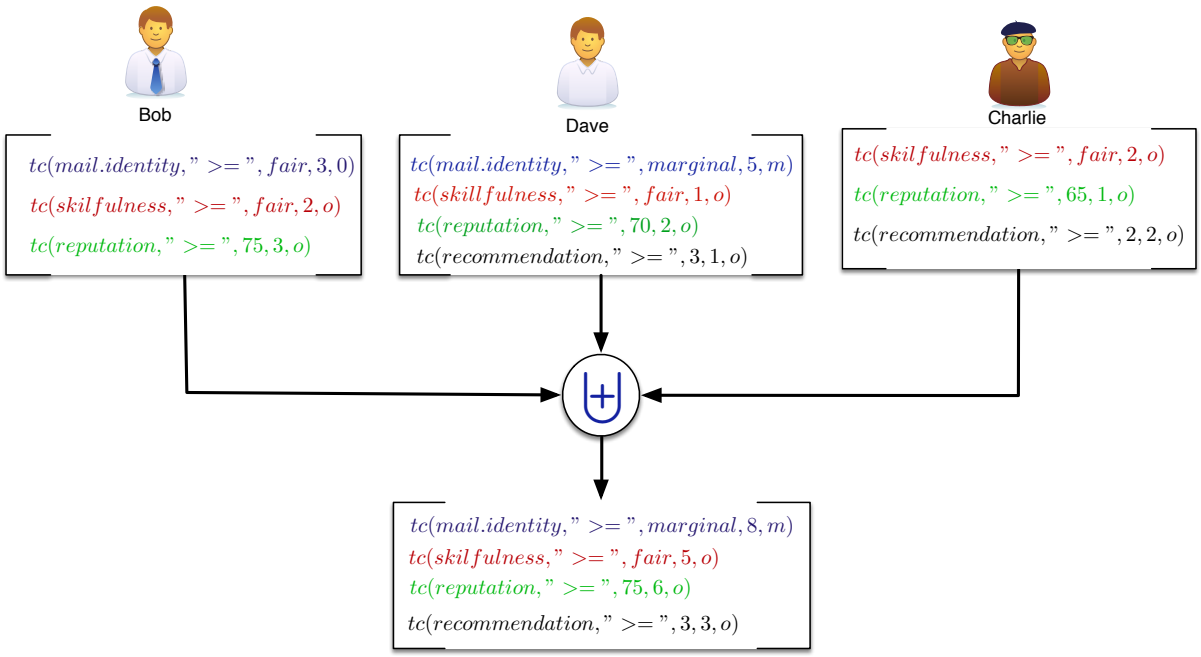

Fig. 3. Illustration of the combination of individual policies

Complex operations allow the creation and/or modification of a policy based on other policies. It is mainly through this type of operations that the collective dimension of trust management operates. Complex operations are defined as follows:

$$\forall \omega_i \in \Lambda^{\Pi},\ \omega_i : 2^{\Pi} \rightarrow \Pi \tag{8}$$

Concretely, two adaptation operators have been defined to enable agents to mimic the *majority social influence* observed by the sociologists [46, 1, 36].

Combine(Π', c, mh, π') operation is used to combine a set of policies $\Pi' = \{\pi_n, \pi_m\}$ (where $n \neq m$ are agents identifiers) into a policy π' representing the collective policy of the community c. So this operator allows a set of agents, structured in a community, to shape a collective policy based on the individual policy of each member. The policies combination mechanism relies on policies composition heuristics [52]. As an example, we illustrate in Figure 3 how this operator is used in this situation to generated a collective policy based on three distinct individual policies. Here, the heuristic is configured to be *reject* conservative. In other words, the generated collective policy will never accept a request that any of the members would have rejected. Analogously, we defined different heuristics that the *combine* operator will use in order to cover all imaginable situations that could exist in real communities (e.g., strong consensus, weak consensus,etc.).

Integrate$(\pi_1, \pi_2, \texttt{ih})$ primitive integrates two policies using the heuristic `ih`. The heuristics here are used in the same way as in the combination. Two policies can be integrated in many distinc ways, each way produces a separate outcome that reflects the intent of the agent performing the integration. For instance, when policies to be integrated are in conflict, the heuristics determines if the TMS should keep the most restrictive value, or the less restrictive ones. Another heuristic could guide the system towards to arbitrate the conflict be proposing a half-way value. In fact, this operator is used by the agent to integrate its individual policy with the collective policy that is applicable in his community. So the integration can be seen as a particular case of *combination* and hence makes use of heuristics that are similar to those discussed in the *combination*.

3.4 Policies Evolution

In addition to *combination* and *integration*, agents are also able to mimic *minority social influence* [36] through collective policies evolution mechanisms. Indeed, for many reasons, any collective policy used within a community may have to become obsolete. For instance, when members turnover pace is very high, new members may no longer feel represented in the effective collective policy, calling for its adaptation. To that aim, we introduced a third adaptation operator (i.e., **Evolve**) to make policies evolution possible. The *Evolve* operator allows each member of the community to adapt it and to propose this new version to the other members. The detailed mechanisms is presented here-after.

Collective policies evolution is performed in the following steps:

The first step is responsible of detecting situations necessitating the adaptation of the collective policy and to trigger the appropriate adaptation. This is performed using a meta-policy that we state as follows:

$$\begin{aligned} &evolve(\pi^{Pattern}, c) : k_0, \ldots, k_n \leftarrow \forall a_i \in c.A \\ &\langle a_j.\varepsilon, inform, a_i.\varepsilon, \\ &cfe(Pattern, Adaptation, TrustFactor)\rangle \end{aligned} \tag{9}$$

In this meta-policy, the context conditions $k_0, \ldots, k_n$ captures the context in which the collective policy has to be adapted. This context can be generic (e.g. performed systematically by a new member) or specific to a particular situation (e.g. after several failures in negotiation because of the same trust criterion). If the context holds, the initiator agent (i.e. a_j) informs its community fellows ($\forall a_i \in c.A$) about the adaptation he wants to perform on the collective policy (i.e. $(Pattern, Adaptation, TrustFactor)$). Here, $Adaptation$ refers to a subset of simple operations ($\forall \omega_i \in \Lambda^{\Pi}$) though which the agent will bring changes to the collective policy.

When a member receives the call for evolution (i.e. $cfe(Pattern, Adaptation, TrustFactor)$), the corresponding meta-policy is triggered to evaluate whether the agent personal conditions to accept such proposal are met. Such meta-policy is stated as follows:

$$\begin{aligned} &cfe(Pattern, Adaptation, TrustFactor) : \\ &k_0, ..., k_m \leftarrow \forall a_i \in c.A \\ &\langle a_j.\varepsilon, inform, a_i.\varepsilon, \\ &vote(agree, Pattern, Adaptation, TrustFactor)\rangle \\ &countVotes(Pattern, Adaptation, TrustFactor) \end{aligned} \tag{10}$$

With the above meta-policy, the agent receiving the call for evolution will accept the call and send an agreement statement that every agent will receive. Of course, the agreement of an agent may influence the agreement of another agent but we do not consider/handle such influence in our work. If the evolution conditions stated by the agent do not hold, the agent will disagree and notify other agents accordingly. Worth noting that the set of conditions used by the agent a_i to trigger the evolution process are different from those used by each agent to accept or not such proposition.

Once the agents have finished voting[6], the evolution initiator and every member of the community count the votes that agree with the proposition and those which do not. Then each

[6] We assume that agents make use of a timer to control the voting process time.

agent makes use of a voting system to determine whether the adaptation has been accepted by the community members or not. This can be done via the following meta-policy:

$$\begin{aligned} &countVotes(Pattern, Adaptation, TrustFactor) : \\ &count(vote(\_, Pattern, Adaptation, TrustFactor)) \\ &\geq (|c.A| * 2/3) \wedge \\ &(count(vote(disagree, Pattern, Adaptation, TrustFactor)) > \\ &count(vote(agree, Pattern, Adaptation, TrustFactor))) \\ &\leftarrow makeEvolution(Pattern, Adaptation, TrustFactor) \end{aligned} \tag{11}$$

This meta-policy counts the total votes and compares it with the community population. If two-thirds of the population have voted, the vote is considered to be valid. When the vote is valid and the agreeing voters outnumber the disagreeing voters, the collective policy is adapted. When the proposition to adapt the collective policy is accepted, each agent (those who agreed and those who did not) makes use of the following meta-policy to make the adaptation of the collective policy effective.

$$\begin{aligned} &makeEvolution(Pattern, Adaptation, TrustFactor) : \\ &policy(P, Pattern) \wedge P.Issuer == c \wedge \\ &Adaptation == \texttt{"restrict"} \\ &\leftarrow RestrictCriterion(P, TrustFactor) \end{aligned} \tag{12}$$

This last meta-policy selects the collective policy that corresponds to the pattern concerned by the adaptation. Then the adaptation operation is performed on the policy. Of course, this meta-policy applies only to the evolutions that makes the collective policy more restrictive. Similarly, other meta-policies are defined for each adaptation operation.

4 System Evaluation

The principal objective of our experiments is to study the impact of the trust management on the dynamics of agents communities. We are particularly interested by the evaluation of the impact of using the concepts drawn from the *social influence theory* (i.e., combination, integration and evolution) on the management of trust withing large systems. To that aim, we used the Recursive Porous Agent Simulation Toolkit (Repast) [12], a free and open-source multi-agent simulation platform, to simulate an open innovation scenario. Our principal motivation in using Repast lies in the fact that this platform is currently considered as the most suitable simulation framework for social scientific agent based computer simulation [20]. This makes Repast the most appropriate candidate for experimenting the benefits of using the Social Influence Theory in Trust Management.

4.1 Open Innovation Scenario

The simulated scenario is based on open innovation communities. Open Innovation is currently recognized as an exciting new way to generate breakthrough innovation at lower cost and fast time. This approach is adopted by companies and organizations to enhance innovation in their R&D departments by harnessing external ideas. Open innovation is often conflated with the *open source* approach for software development. Indeed, both approaches rely on the abundance

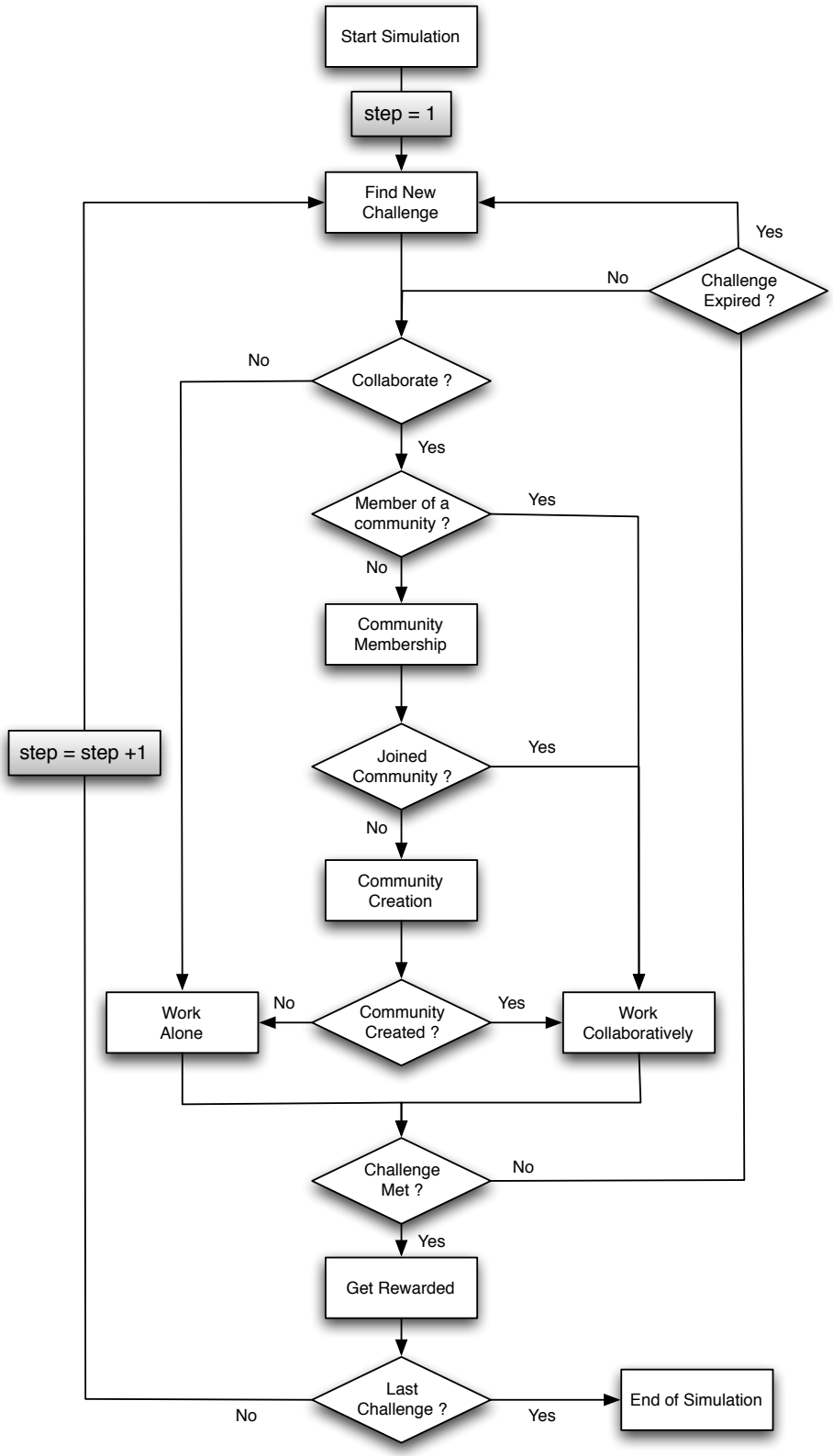

Fig. 4. Simulation Process

of external source of idea to create value. However, open innovation incorporate the business model in the innovation process, while *open source* approach downplays or denies it [22]. With respect to this issue, participants in *open innovation* process are more concerned by *trust* as the business model represents the source of both value creation and capture.

4.2 Simulation Model Description

In our scenario, we consider a system that is composed of a set of agents that try to solve the same problem. Collaboration allows these agents to solve the proposed problems more quickly, maximizing their chance to gain the reward. For simplicity, we assume that one participant is playing the role of *seeker* and the other participants are playing the role of *solver*. The simulation is initialized by creating one *seeker* and a given number of *solvers*. The objective of the *seeker* is to find a solution to a given number of problems (i.e. `chCount`), while solvers aspire to optimize their individual utility. This utility can be increased by winning a challenge and obtaining the reward, or by being a member of a community that won a challenge. A challenge is a problem to which the seeker is willing to give a reward against a solution that satisfies its requirements. Defining how a solution is considered satisfactory by the seeker is out of the scope of this work.

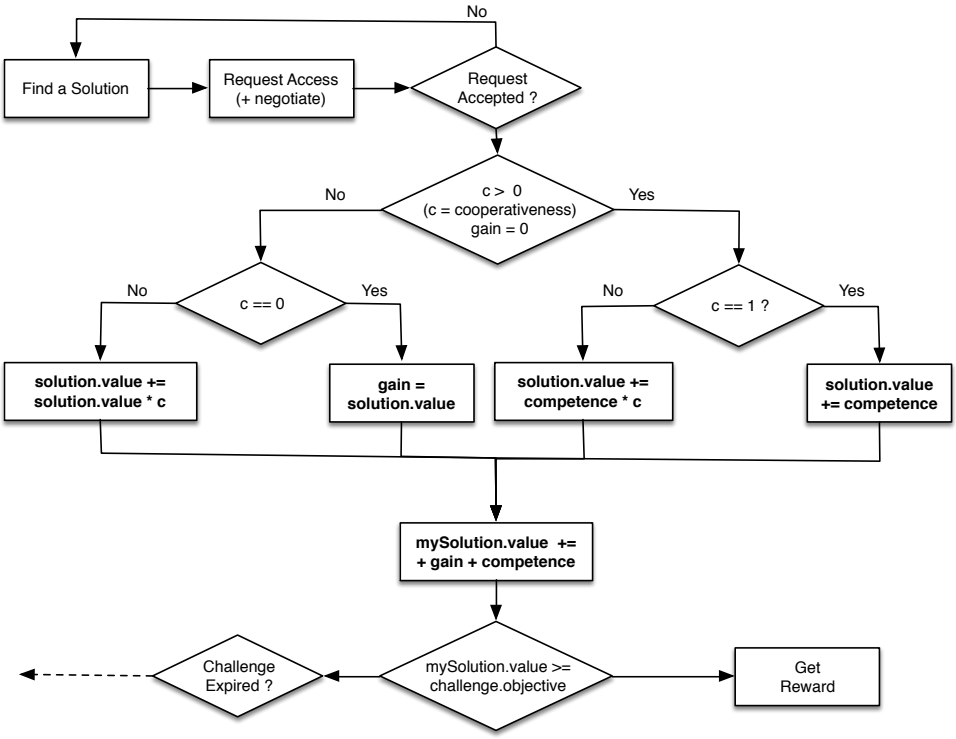

Fig. 5. Flowchart describing the abstract innovation process

4.3 Open Innovation Simulation

The simulation starts when a new challenge is made available by a *seeker*, as illustrated in Figure 4. Once a new challenge is found, the *solver* decides whether to *work collaboratively* or to *work alone* on it. This decision is defined by its propensity to collaborate that we express as a probability drawn from a uniform distribution $\mathcal{U}[0,1]$ (i.e.*collaborativeness*). If the *solver* decides to collaborate, he first checks whether he is not already member of a community. If not, the *solver* will try to join a community otherwise, he will try to create a new one.

At this stage, if the member succeed to join or create a community, he will *work collaboratively*, otherwise, he will *work alone*. A round of this simulation model is completed when every *solver* has been given exactly one opportunity to work. Once the *solver* finishes working, he evaluates whether the value of the solution he participated in reached the *objective* of the challenge. If it is the case, the state of the challenge is set to *met* and the agent (or community of agents) is rewarded. If not, the *seeker* checks whether the challenge did not expire, otherwise he introduces a new one. If the *seeker* reaches the number of challenges he wanted to solve, the simulation stops.

In this scenario, two types of trust decisions are made. The fist one concerns the *admission of news members* in the community, and the second one about the *access to the resources* encapsulating the solutions. For each type of decision, we defined two policies, an individual one, specific to each agent and a collective one, created withing the community that the members have to comply with. We make use of norms to define sanctions agents have to face in case of non compliance [52]. In addition, all decisions made by any agent are systematically verified by the Simulation System[7]. Each time an agent takes a decision against the collective policy, this agent is sanctioned. After three sanctions, the agent is evicted from the community. We manage this process of sanction and eviction using the norms defined in the $\mathcal{M}oise$ specification of community.

4.4 Trust Decision Making Process

Up to now, we described the process followed by the *solver* during the simulation in abstract terms. For instance, we did not detail what is really happening in the working (alone or collaboratively) step. To that aim, we present in Figure 5 how the *cooperativeness* attribute influences the behavior of solvers.

As illustrated in the figure above, if the *cooperativeness* value is positive and not null (i.e. > 0), the *solver* will improve the *solution* (i.e. `x.solution`) to which he is accessing proportionally to its *cooperativeness*. Contrariwise, if this value is null or negative, the solver will profit from the solution and vandalize it proportionally to its *cooperativeness* value. *Rewards* are determined in terms of utility. At the beginning of the simulation, each *solver* is provided an initial utility score of 100 units. Then this score is affected (positively or negatively) by the decisions made by this *solver*.

A *solver* can gain utility units by winning a challenge or participating in a community which won a challenge. A *solver* can also gain utility when accessing valuable resources or by letting *competent* and *cooperative solvers* manipulate its resources, or the resource of its community.

A *solver* can also loose *utility* units. The *utility* of a *solver* decreases in three situations: (1) when his *solution* has been vandalized by a malicious *solver*, (2) when the solution of its community has been vandalized, and (3) when he violates the collective policy of its community.

4.5 Experimentation parameters

The simulation model is essentially composed of two type of agents (*seeker* and *solver*), a class encapsulating the concept of community, a class representing the solutions proposed by *solvers* and a class representing the challenge. In this section, we present the parameters used to setup our simulation. The parameters of the experimentation fall into two categories; parameters that relate to the challenges and those that relate to the solvers. These parameters represent default settings if not stated otherwise:

[7] For simulation purposes, we created a Big Brother component that is able to observe all interactions and their outcome. It is this component that extracts simulation metrics and computes statistics.

	Attribute	Values	Description
Challenge	`Objective`	`[0, 10^4]`	Corresponds to the value that a solution should reach to fulfil the challenge
	`Reward`	`[0,10^3]`	Corresponds to the utility the agent(s) that proposed the solution will gain in terms of utility. If the challenge is fulfilled by the members of a community, they will share the utility.
	`Deadline`	`[0,10^3]`	Corresponds to the number of simulation steps that the challenge will remain active before its expiration.
Solver	`competence`	$\mathcal{N}(0,1)$	Corresponds to the solver's competence degree. It determines to which extent the solver is able to improve a solution.
	`collaborativeness`	$\mathcal{U}[0,1]$	Expresses to which extent a solver is willing to work collaboratively. It represents the probability to see the solver creating/joining a community.
	`cooperativeness`	`[-1,1]`	Determines how the solver behaves when accessing resources that he does not own.
	`interation`	`0.8`	Corresponds to the provability of an agent to have an interaction.

Table 1. The variables of a challenge

- *Challenges'* parameters:
  1. `Challenges Count`: the number of challenges introduced.
  2. `Challenges Objectives`: what is the objective (i.e. resource value) that the agents much reach to win the challenge. The objective of each challenge was set to 10^4.
  3. `Challenges Rewards`: what is the reward that agents will gain in winning the challenge. The reward of each challenge was set to 1000.
  4. `Challenges Deadline`: what is the deadline of the challenge. The deadline of each challenge was set to 1000.
- *Solvers'* parameters:
  1. `Policies`: each solver is endowed with two policies which *type*, *value* and *weight* are randomly generated. The first policy is used for making decisions about access to resources, while the second is used to make decisions about whom to collaborate with (i.e. community membership).
  2. `Competence`: the *competence* of *solvers* is drawn from a *centered normal* distribution $\mathcal{N}(0,1)$.
  3. `Collaborativeness`: the *collaborativeness* of *solvers* is drawn from a uniform distribution $\mathcal{U}[0,1]$.
  4. `Cooperativeness`: the *cooperativeness* of solvers was set manually in order to have a population that is composed of 10% altruist, 40% cooperative, 40% selfish and 10% malicious solvers.
  5. `Interaction`: this parameter determines the probability of a solver to interact with other solves (i.e. request access to solution). This parameter has been fixed to 0.8.

4.6 Simulation metrics

Metrics represent the statistics that we rely on in the evaluation of our approach. These metrics are presented in what follows:

- `sovCount`: the number of solvers in the system.
- `comsCount`: the number of communities
- `comsSize`: the population of each community

The table 1 summarizes the parameters presented above.

4.7 Hypothesis and Results

The innovative aspect of our trust management model lies in the joint use of the collective and individual dimensions during trust assessment. This joint consideration is made possible using policies adaptation mechanisms through policies *combination*, *integration* and *evolution*. Therefore, in our evaluation, we get particularly focused on the assessment of the impact of these mechanisms on the dynamics of agents communities.

In our experiments, we will rely on the number of communities and the mean size of each community to grasp it. We assume that systems with bigger number/size of communities constitute a better place for collaboration and thus innovation. We assume also that the more a community is bigger the less it is prone to collapse.

In this evaluation, we used 1000 *solvers* and 1 *seeker* which was initialized with 10 challenges. Each challenge has a deadline of 1000 simulation steps, thus the average duration of a simulation is around 10000 steps. The objective has been fixed with respect to the standard settings presented previously. Also, communities do not necessary explode after the end of each challenge as long as a new challenge is rapidly introduced. Members prefers to remain in a community rather then leaving it.

We simulated seven experimental populations. A summary of the parameters used for each population is provided in 2. Populations have different behaviors with respect to our model (combination, integration and evolution). Moreover, we used populations with different degrees of integration in order to assess the impact of the compliance on the dynamics of communities. Also, in our scenario, we make use of random draws which make our simulation *stochastic*. As a *stochastic* simulation, our evaluation will produce different outcomes for different random number streams, which are generally driven by choosing different random seeds. For instance, some *solvers* may interact only with good *solvers* (i.e. *altruist* or *cooperative*) or bad *solvers* (i.e. *selfish* or *malicious*) *solver*. In addition, the result of each *solver* is also the consequence of the values they have been initialized with. So in order to minimize this effect on our results, and to explore the space of all possible outcomes, all graphs presented plot the mean of 100 execution of the simulation. Then we compared the results of each population and that we discuss hereafter.

population	Combination	Integration	Evolution
population 1	`false`	0	`false`
population 2	`true`	0	`false`
population 3	`true`	0.5	`false`
population 4	`true`	0.8	`false`
population 5	`true`	1	`false`
population 6	`true`	0.8	`true`
population 7	`true`	1	`true`

Table 2. Populations used in the simulations

With respect to this settings, we formulated *four working hypotheses* that we aim to demonstrate.

Hypothesis 1 Here we want to show that *communities wherein agents are able to perform policies combination in order to emerge a collective policy have a better dynamic than communities composed of "non combining" agents*.

In this simulation, *population 1* is used for control purpose, while *population 2* is used to assess the benefit of combination.

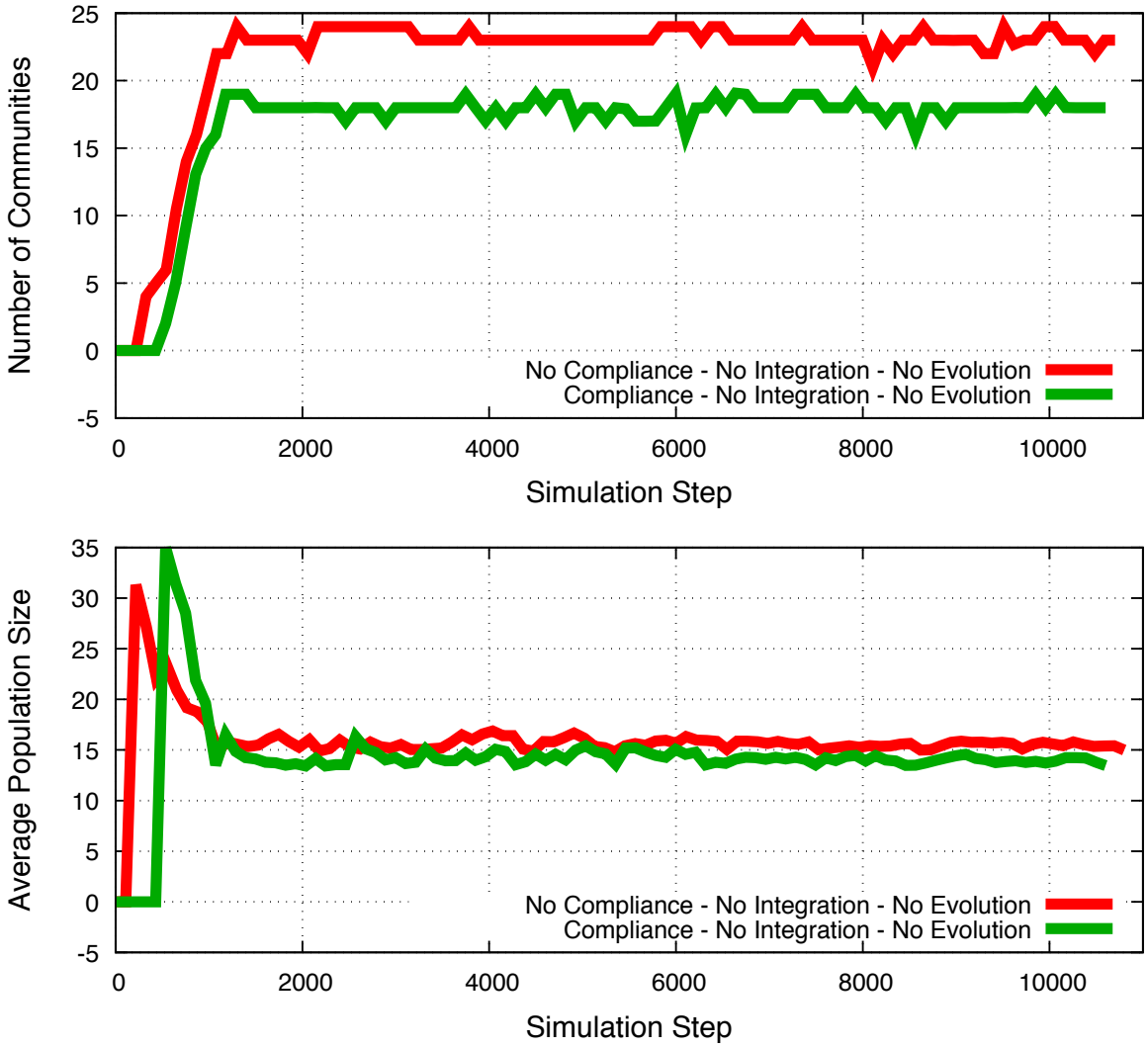

Fig. 6. Evaluation of Hypothesis 1

As showed in Figure 6, the effect of *combination* on *population 2* was quite negative. We explain this result by the fact that agents from *population 2* are handicapped by the process of building collective policies while they do not benefit from it as they are unable to integrate theses policies. Thus without integration mechanisms, this feature will only have a prescribing effect, making agents more frequently sanctioned (excluded), and communities collapsing more often. Also, we observe that all agents (60%) do not necessary belong to a community. This is due to their willingness to collaborate (to be in communities) and also to their ability to trust each other.

Hypothesis 2 This hypothesis considers *the dynamics of communities composed of compliant agents (able to integrate collective policies) are better than the dynamics of communities composed of non compliant agents*. Here we try to verify the impact of the combination where agents are able to, not only emerge collective policies from individual ones (i.e., combination), but also to comply with theses policies through integration. For this purpose, we used *population 5* in the same settings that we used to simulate *population 1* and *population 2*. The results of these evaluations are plotted in Figure 7.

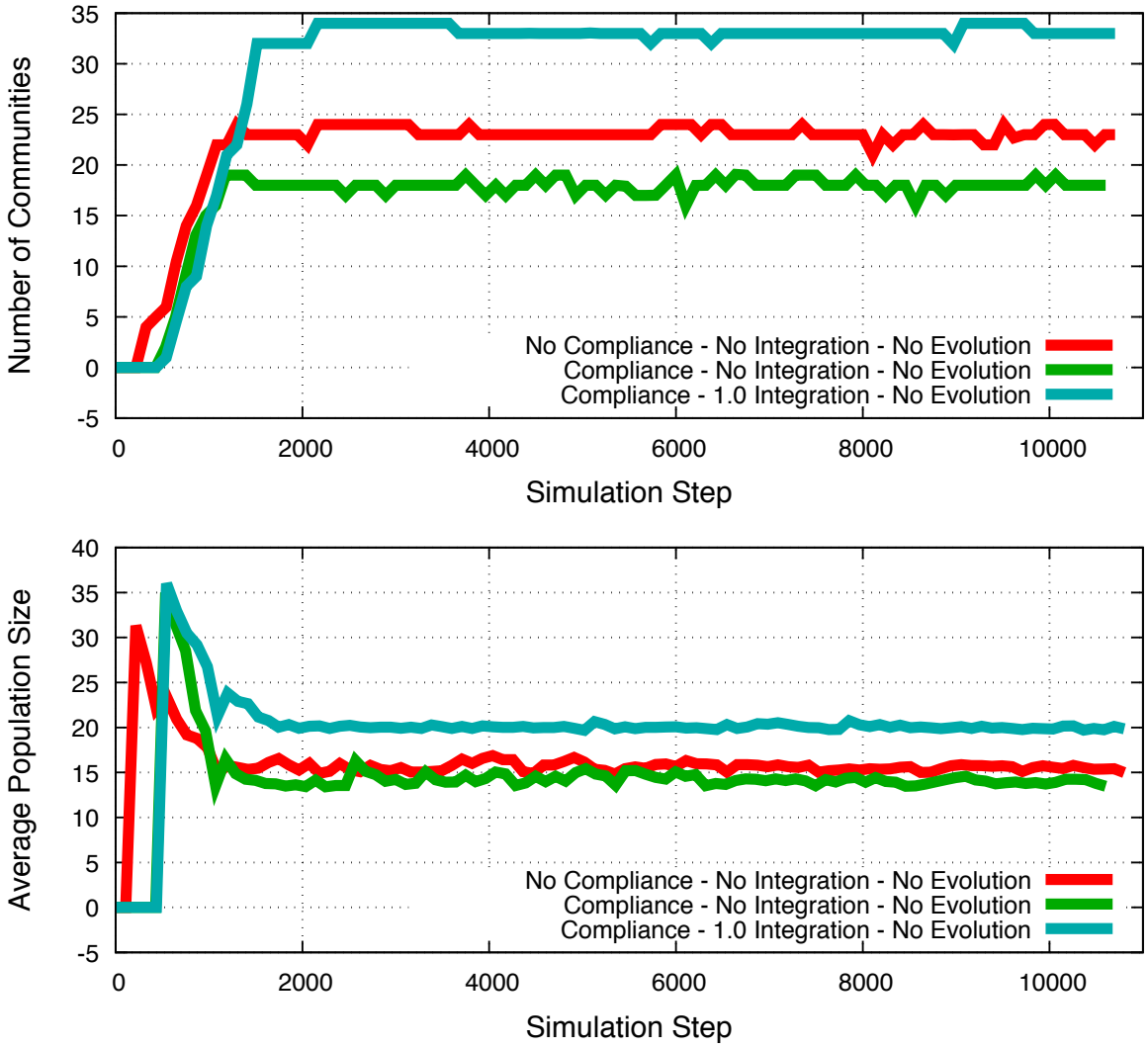

Fig. 7. Evaluation of Hypothesis 2

As expected, *populations 5* produces the best results. The communities created in this population have a much better dynamic as their members systematically comply with collective policies. To simplify, agents of this population never violate their collective policies and thus are never excluded, making communities collapse less frequent.

Hypothesis 3 Here we want to check that *communities dynamics is function of the proportion of* compliant *members composing it*. In the previous experiment, we showed that 100% compliant communities reveal to be more stable and less prone to *population* turnover. We explained that by the fact that their members are never sanctioned and thus, never excluded. However, the *social impact theory* explains that such *population* never exists in real settings. *Ash* [1] reported in his experiences a proportion of 80% of *compliant* individuals in real life settings. In the light of that, we make use, in this simulation, of *population 3* and *population 4* which possess different proportion of *compliant* solvers in order to see if these communities are able to maintain positive dynamics in presence of non-compliant agents. The results showed that *communities* stability is positively affected by the amount of *compliant* agent composing the population. This result also reinforced the findings of the previous section in the sense that the propensity of *solvers*

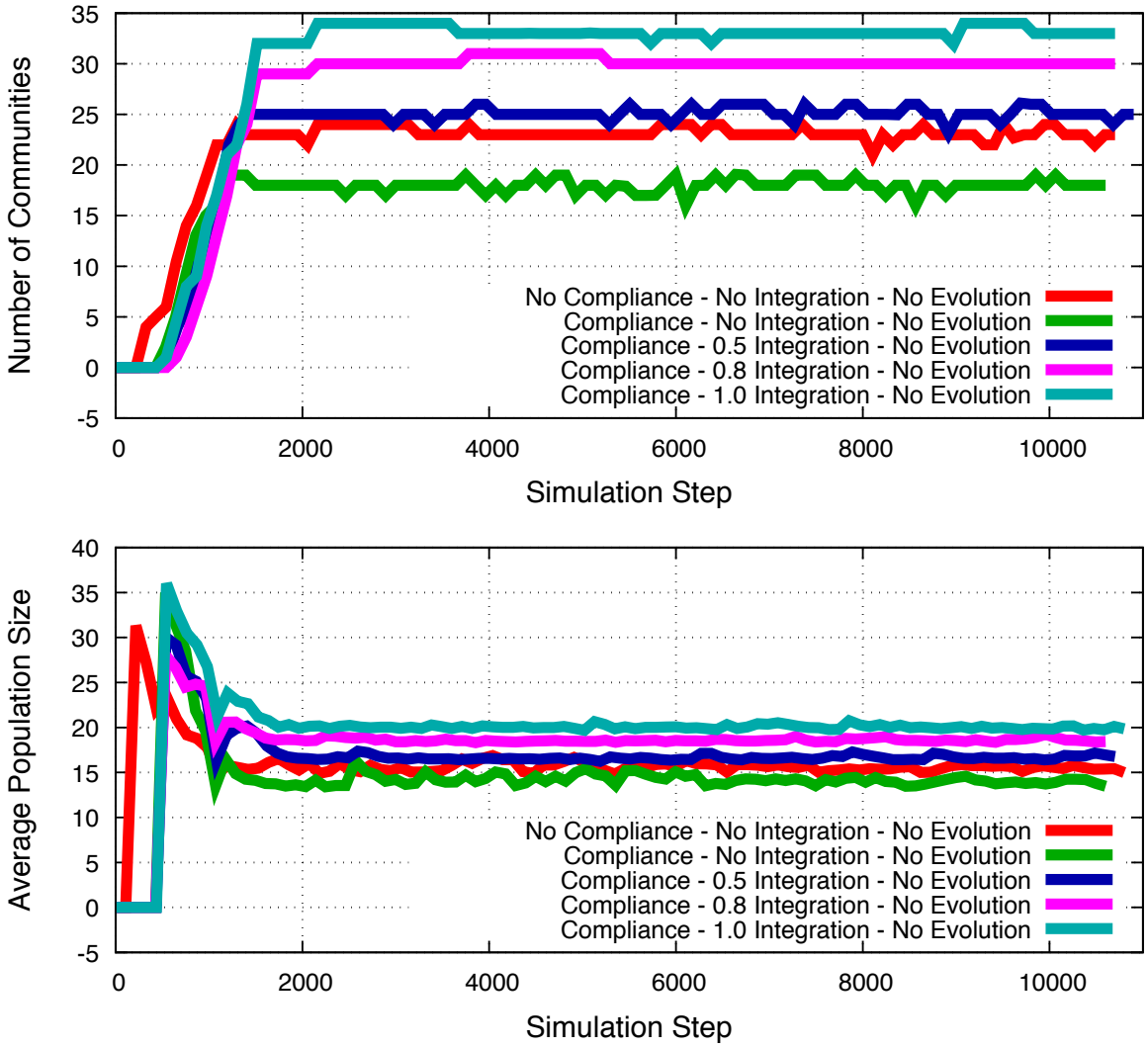

Fig. 8. Evaluation of Hypothesis 3

to comply with the collective policies of their communities affects significantly the stability of their communities.

Hypothesis 4 Here we hypothesize that *The agents ability to make evolve their collective policies affects positively the dynamics of these communities*.

In previous sections, *population 5* reveals to be the *population* sample having the most stable communities. However, the experiments conducted in these sections showed also that this population was not so idyllic as one could expect. As discussed so far, a *solver* can leave a community because it has been *excluded* after having violated the collective policy, or because of a too restrictive or a too permissive collective policy. The first situation is not relevant for *population 5* as the *solvers* of this population never violate their collective policies. Thus, the solely explanation that we can defend is that these *solvers* leave their communities because they enter in conflict with the *collective policies* in use in their communities. It is this issues that we will investigate in this section. Our objective is to see whether putting in place the complete *social influence* micro-macro (top-down and Bottom-up) loop will reduce the number of agents leaving their population, and thus participate in the improvement of communities stability.

For this purpose, we make use of *population 6* and *population 7* aforementioned. These populations correspond to, respectively, *population 4* and *population 5* in which the *evolution* features have been activated. So the *solvers* of both populations are able to trigger evolution of the collective policies of their communities. The results of these populations are plotted in Figure 9. As shows in Figure 9, after allowing *solvers* to change their collective policies, the number

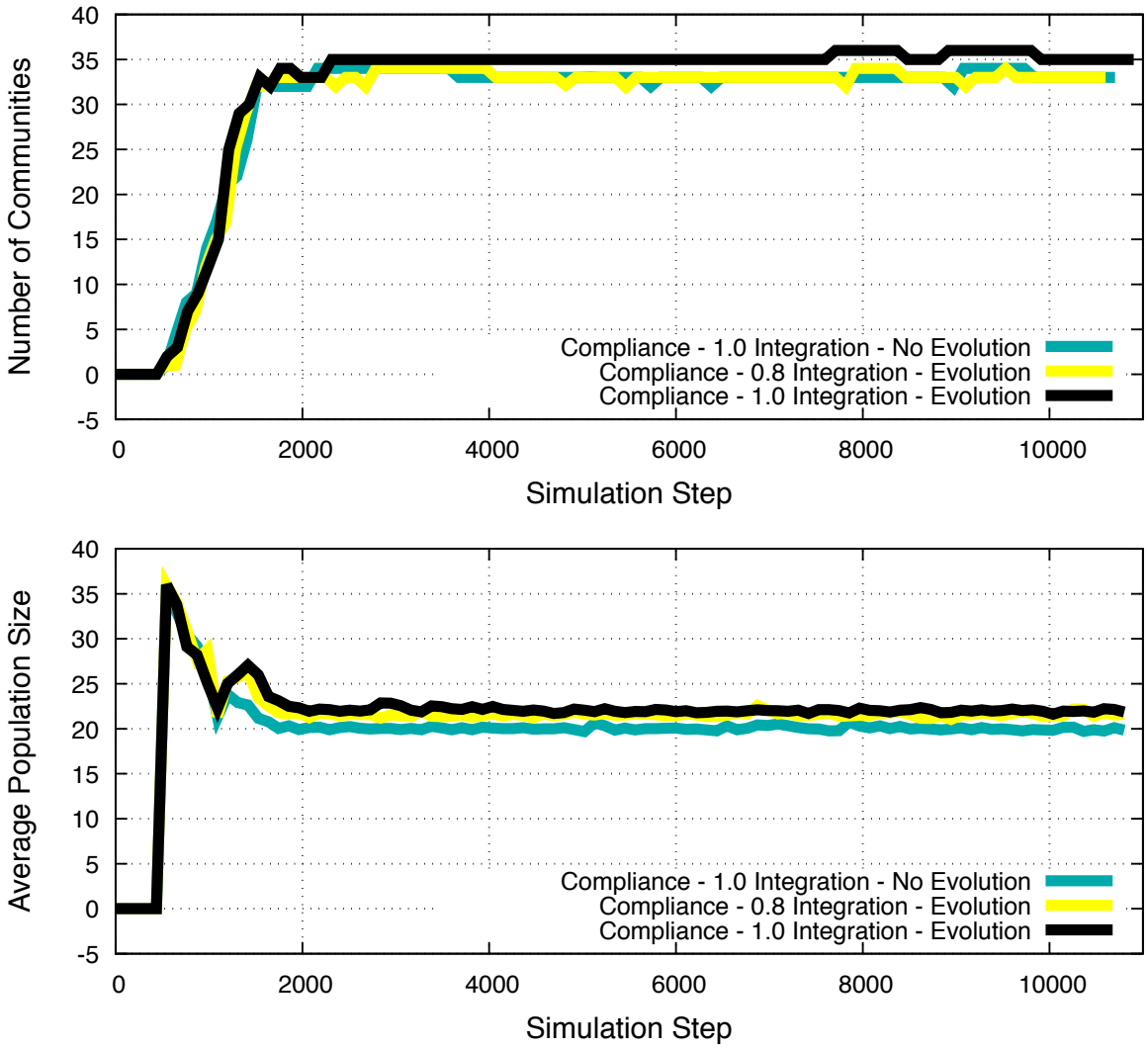

Fig. 9. Evaluation of Hypothesis 4

of communities in *population 4* (6 in this evaluation) had risen above 30, while this number exceeded 35 for *population 5* (7 in this evaluation). These results showed that *populations* in which only 80% of *complying solvers* reveal to be able to have as much stable communities as *communities* which members were 10% complying.

The analysis of the above results in terms of *communities size* revealed that the *evolution* feature reduces substantially the members turnover. Surprisingly, *population 6* (former 4) was even able to perform better that *population 5* as the average population of each community was almost as good as in *population 7*. While, our experiments do not allow to explain these effects, we thinks that the *ability* of solvers to adapt their collective policies makes improves their stability. This stability participates in the growth of the size of these communities, and makes these *communities* scale.

In sum, based on the above evaluation, we assume that *evolution* has a positive impact on the stability and even *scalability* of virtual communities. These results confirmed also the benefit of applying *social influence theory* to trust management as *socially compliance solvers* tend to form more stable and bigger communities.

5 Conclusion

We studied the impact of a so called *collective and socially-aware management of trust* within self-organized and self-governed communities of agents. The principal objective of our study was to demonstrate the impact of such approach could have on the dynamics of the communities in term of number and size. To that aim, we first presented the model we used to adapt individual and collective policies. To build this model, we draw inspiration from social influence phenomena [52] observed by several sociologists in the context of the social influence theory [46, 1, 36, 28]. Furthermore, we defined a simulation scenario using open innovation paradigm in order to evaluate our approach.

The results that have been presented and discussed showed that by allowing agents to adapt their collective policies (i.e., evolution), we can significantly stimulate the dynamics of communities. We make the hypothesis that such a positive communities dynamics (i.e., increase in the number and size of populations) promote the stability of these communities and stimulate the collaboration.

Although our evaluations show that our approach has a reasonably positive impact on the *dynamics*, and hopefully the stability, of virtual communities, several questions remain open and motivate for further investigations. For example, a basic question is "what would be the impact of the model on populations composed of only *malicious* or *selfish* participants, or different mixtures of such profiles?". Indeed, in our experiments we showed that the ability of agent to adapt their collective policies affects positively the stability of these communities. However, we did not evaluate situations in which malicious members collude to make the collective policy less restrictive. We believe also that it is worth verifying if the benefits brought by our model do not have any negative counterpart in terms of performances or any impact on the efficiency of agents. Such questions call for further in depth evaluations to assess the performances of our model in different settings.

Bibliography

[1] S. E. Asch. Opinions and Social Pressure. *Scientific American*, 193:31–35, 1955. ISSN 00368733. doi: 10.1038/scientificamerican1155-31.

[2] E. Bertino, E. Ferrari, and A. Squicciarini. X -tnl: An xml-based language for trust negotiations. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY '03, pages 81–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1933-4. URL http://dl.acm.org/citation.cfm?id=826036.826848.

[3] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-x: A peer-to-peer framework for trust establishment. *IEEE Trans. on Knowl. and Data Eng.*, 16(7):827–842, July 2004. ISSN 1041-4347. doi: 10.1109/TKDE.2004.1318565. URL http://dx.doi.org/10.1109/TKDE.2004.1318565.

[4] Thomas Blass. The Milgram Paradigm After 35 Years: Some Things We Now Know About Obedience to Authority1. *Journal of applied social psychology*, pages 955–978, 1999. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1559-1816.1999.tb00134.x/abstract.

[5] M. Blaze, J. Feigenbaum, and A. D. Keromytis. Keynote: Trust management for public-key infrastructures. In Bruce Christianson, Bruno Crispo, WilliamS. Harbison, and Michael Roe, editors, *Security Protocols*, volume 1550 of *Lecture Notes in Computer Science*, pages 59–63. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65663-0. doi: 10.1007/3-540-49135-X_9. URL http://dx.doi.org/10.1007/3-540-49135-X_9.

[6] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 164–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7417-2. URL http://dl.acm.org/citation.cfm?id=525080.884248.

[7] P.A. Bonatti, L Juri, Daniel Olmedilla, and Luigi Sauro. Policy-driven negotiations and explanations: Exploiting logic-programming for trust management, privacy & security. *Logic Programming*, pages 779–784, 2008. URL http://link.springer.com/chapter/10.1007/978-3-540-89982-2_76.

[8] Piero Bonatti and Pierangela Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 134–143, New York, NY, USA, 2000. ACM. ISBN 1-58113-203-4. doi: 10.1145/352600.352620. URL http://doi.acm.org/10.1145/352600.352620.

[9] Piero Bonatti, J. L. De Coi, Daniel Olmedilla, and Luigi Sauro. A rule-based trust negotiation system. *IEEE Trans. on Knowl. and Data Eng.*, 22(11):1507–1520, November 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2010.83. URL http://dx.doi.org/10.1109/TKDE.2010.83.

[10] Piero A. Bonatti and Pierangela Samarati. Regulating Service Access and Information Release on the Web. *Journal of Computer Security*, 10(3):241 – 271, 2002.

[11] L.M. Camarinha-Matos, O. Castolo, and J. Rosas. A multi-agent based platform for virtual communities in elderly care. In *Proceedings of 2003 IEEE Conference on Emerging Technologies and Factory Automation.*, volume 2, pages 421–428. IEEE, 2003. ISBN 0-7803-7937-3. doi: 10.1109/ETFA.2003.1248730. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1248730.

[12] N. Collier. RePast : An Extensible Framework for Agent Simulation, 2003.

[13] J.L.T. da Silva and Y. Demazeau. Vowels co-ordination model. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, AAMAS '02, pages 1129–1136, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0. doi: 10.1145/545056.545083. URL http://doi.acm.org/10.1145/545056.545083.

[14] Morton Deutsch and Harold B. Gerard. A study of normative and informational social influences upon individual judgment. *The Journal of Abnormal and Social Psychology*, 51 (3):629–636, 1955. ISSN 0096-851X. doi: 10.1037/h0046408. URL http://doi.apa.org/getdoi.cfm?doi=10.1037/h0046408.

[15] Rino Falcone and Cristiano Castelfranchi. Social trust: a cognitive approach. In Christiano Castelfranchi and Yao-Hua Tan, editors, *Trust and deception in virtual societies*, pages 55–90. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0-7923-6919-X. URL http://dl.acm.org/citation.cfm?id=379153.379157.

[16] T. W. A. Grandison. *Trust Management for Internet Applications*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 2003. URL http://www.doc.ic.ac.uk/~mss/Papers/Grandison-phd.pdf.

[17] Tyrone Grandison and Morris Sloman. Trust management tools for internet applications. In *Proceedings of the 1st international conference on Trust management*, iTrust'03, pages 91–107, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40224-1. URL http://dl.acm.org/citation.cfm?id=1759008.1759015.

[18] S. Gupta and H.W. Kim. Virtual community: concepts, implications, and future research directions. In *Proceedings of the 10th American Conference on Information System*, pages 2679–2687, 2004.

[19] Errol E Harris. Kants Refutation of the Ontological Proof. *Philosophy*, 52:90, 1977. ISSN 00318191.

[20] Robert Tobias Hofmann and Carole. Evaluation of free Java-libraries for social-scientific agent based simulation, jan 2004. URL http://jasss.soc.surrey.ac.uk/7/1/6.html.

[21] J. F. Hubner, J. S. Sichman, and O. Boissier. MOISE+ : Towards a structural , functional , and deontic model for MAS organization. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1–2. ACM Press, 2002. ISBN 1581134800.

[22] Eelko K R E Huizingh. Open innovation: State of the art and future perspectives. *Technovation*, 31:2–9, 2011. ISSN 01664972. doi: 10.1016/j.technovation.2010.10.002.

[23] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2): 119–154, September 2006. ISSN 1387-2532. doi: 10.1007/s10458-005-6825-4. URL http://dx.doi.org/10.1007/s10458-005-6825-4.

[24] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, volume 160, pages 324–337, 2002. doi: 10.1.1.60.5461.

[25] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 402–418. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20362-9. doi: 10.1007/978-3-540-39718-2_26. URL http://dx.doi.org/10.1007/978-3-540-39718-2_26.

[26] A. A. Kalam, S. Benferhat, A. Miège, R. El Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization based access control. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*,

POLICY '03, pages 120–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1933-4. URL http://dl.acm.org/citation.cfm?id=826036.826869.

[27] Immanuel Kant. *Kritik der praktischen Vernunft*, volume 56. Gutenberg Project, 1788. ISBN 3518076566.

[28] B. Latané. The psychology of social impact. *American Psychologist*, 36:343–356, 1981.

[29] Niklas Luhmann. Familiarity, confidence, trust: Problems and alternatives. In *Trust: Making and breaking cooperative relations*, pages 15–35. Basil Blackwell, 1990.

[30] JT Mahoney, AS Huff, and JO Huff. Toward a new social contract theory in organizatioon science. *Journal of Management Inquiry*, 1994. URL http://jmi.sagepub.com/content/3/2/153.short.

[31] D.W. Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *Proceedings of 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*, pages 312–321. IEEE Comput. Soc, 1998. ISBN 0-8186-8292-2. doi: 10.1109/ICDCS.1998.679731. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=679731.

[32] Stephen Paul Marsh. *Formalising trust as a computational concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, 1994. URL http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Formalising+Trust+as+a+Computational+Concept\#0.

[33] Daniel J. Mcallister. Affect- and Cognition-Based Trust as Foundations for Interpersonal Cooperation in Organizations. *The Academy of Management Journal*, 38(1):24–59, 1995.

[34] Daniel J. Mcallister. The Second Face of Trust: reflections on the Dark Side of Interpersonal Trust in Organizations. *Research on Negotiation in Organizations*, 6:87–111, 1997.

[35] Barbara Misztal. *Trust in Modern Societies: The Search for the Bases of Social Order*. Wiley, 1996.

[36] S. Moscovici. Studies in Social Influence. *Journal of Experimental Social Psychology*, 16: 270–282, 1969.

[37] Wolfgang Nejdl, Daniel Olmedilla, and Marianne Winslett. Peertrust: Automated trust negotiation for peers on the semantic web. In Willem Jonker and Milan Petković, editors, *Secure Data Management*, volume 3178 of *Lecture Notes in Computer Science*, pages 118–132. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22983-4. doi: 10.1007/978-3-540-30073-1_9. URL http://dx.doi.org/10.1007/978-3-540-30073-1_9.

[38] C.J. Nemeth. Differential contributions of majority and minority influence. *Psychological review*, 1986. URL http://psycnet.apa.org/journals/rev/93/1/23/.

[39] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, POLICY '02, pages 50–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1611-4. URL http://dl.acm.org/citation.cfm?id=863632.883495.

[40] Plato. *The Republic*. The Gutenberg Project, 1994.

[41] Julian B. Rotter. A new scale for the measurement of interpersonal trust. *Journal of Personality*, 35(4):651–665, December 1967. ISSN 0022-3506. doi: 10.1111/j.1467-6494.1967.tb01454.x. URL http://doi.wiley.com/10.1111/j.1467-6494.1967.tb01454.x.

[42] M. Rupert, S. Hassas, C. Li, and John Sherwood. Simulation of Online Communities Using MAS Social and Spatial Organisations. *World Academy of Science*, pages 355–360, 2007.

[43] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall,, 2010. ISBN 0136042597, 9780136042594.

[44] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, pages 194–195, New York, NY, USA, 2001. ACM. ISBN 1-58113-326-X. doi: 10.1145/375735.376110. URL http://doi.acm.org/10.1145/375735.376110.

[45] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996. doi: 10.1109/2.485845. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485845.

[46] M. Sherif. The psychology of social norms. *Journal for the Theory of Social Behaviour*, 41:53–76, 1936. ISSN 00218308. doi: 10.1111/j.1468-5914.2011.00472.x.

[47] A. Squicciarini, E. Bertino, Elena Ferrari, F. Paci, and B. Thuraisingham. Pp-trust-x: A system for privacy preserving trust negotiations. *ACM Trans. Inf. Syst. Secur.*, 10(3), July 2007. ISSN 1094-9224. doi: 10.1145/1266977.1266981. URL http://doi.acm.org/10.1145/1266977.1266981.

[48] L. Vercouter and G. Muller. L.i.a.r.: Achieving social control in open and decentralised multi-agent systems. *Journal Applied Artificial Intelligence*, 24(8):723–768, September 2010. ISSN 0883-9514. doi: 10.1080/08839514.2010.499502. URL http://dx.doi.org/10.1080/08839514.2010.499502.

[49] Laurent Vercouter and Guillaume Muller. L.I.A.R.: Achieving Social Control in Open and Decentralized Multiagent Systems. *Applied Artificial Intelligence*, 24(8):723–768, September 2010. ISSN 0883-9514. doi: 10.1080/08839514.2010.499502. URL http://dl.acm.org/citation.cfm?id=1858344.1858345.

[50] Oliver E. Williamson. Calculativeness, trust, and economic organization. *Journal of Law and Economics*, 36(1):453–486, 1993.

[51] W. H. Winsborough and N. Li. Safety in automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 9(3):352–390, August 2006. ISSN 1094-9224. doi: 10.1145/1178618.1178623. URL http://doi.acm.org/10.1145/1178618.1178623.

[52] R. Yaich, O. Boissier, G. Picard, and P. Jaillon. Adaptiveness and Social-Compliance in Trust Management within Virtual Communities. *Web Intelligence and Agent Systems (WIAS), Special Issue: Web Intelligence and Communities*, 11(4), 2013.

[53] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1):1–42, February 2003. ISSN 10949224. doi: 10.1145/605434.605435. URL http://portal.acm.org/citation.cfm?doid=605434.605435.