Noname manuscript No.
(will be inserted by the editor)

Conviviality-Driven Access Control Policy

Donia El Kateb · Nicola Zannone · Assaad Moawad · Patrice Caire · Grégory Nain · Tejeddine Mouelhi · Yves Le Traon

Received: date / Accepted: date

Abstract Nowadays many organizations experience security incidents due to unauthorized access to information. To reduce the risk of such incidents, security policies are often employed to regulate access to information. Such policies, however, are often too restrictive, and users do not have the rights necessary to perform assigned duties. As a consequence, access control mechanisms are perceived by users as a barrier and thus bypassed, making the system insecure. In this paper we draw a bridge between the social concept of conviviality and access control. Conviviality has been introduced as a social science concept for ambient intelligence and multi-agent systems to highlight soft qualitative requirements like user-friendliness of systems. To bridge the gap between conviviality and security, we propose a methodological framework for updating and adapting access control policies based on conviviality recommendations. Our methodology integrates and extends existing techniques to assist system designers in the derivation of access control policies from socio-technical requirements of the system, while taking into account the conviviality of the system. We illustrate our framework using the Ambient Assisted Living (AAL) use case from the HotCity of Luxembourg.

D. El Kateb
Laboratory of Advanced Software SYstems (LASSY)
University of Luxembourg
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
E-mail: donia.elkateb@uni.lu

N. Zannone
Eindhoven University of Technology
E-mail: n.zannone@tue.nl

A. Moawad · P. Caire · G. Nain · T. Mouelhi · Y. Le Traon
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
E-mail: assaad.moawad@uni.lu
E-mail: patrice.caire@uni.lu
E-mail: gregory.nain@uni.lu
E-mail: tejeddine.mouelhi@uni.lu
E-mail: yves.letraon@uni.lu

1 Introduction

Building a secure socio-technical system requires the deployment of appropriate security mechanisms [49]. To avoid security incidents due to an unauthorized access, data and IT resources are usually protected by means of access control policies. Access control policies are often defined on the basis of the *least privilege* principle. According to this principle, users should only be able to access the minimum amount of information necessary to accomplish their duties [60]. However, to minimize the risk of unauthorized access, policy designers tend to define policies which are even more restrictive than what is recommended by the least privilege principle. Thereby, access control is often perceived by users as a barrier, an unacceptable limitation. Worse, this barrier is considered as an obstacle that should be bypassed.

There are several real-life examples that show how users try to bypass security mechanisms (including the access control mechanism) that interfere and cause inconveniences [64]. For instance, the employees of a company can give their credentials to consultants in order to allow them access to specific applications. This behavior is clearly against the access control policy and even increases the risks of security breaches, since the sharing of credentials does not make it possible to trace the access to the users who actually accessed a certain application. Moreover, this behavior can lead to several other security problems, such as role usurpation. Thus, the aspiration to make the system more secure actually makes the system more insecure. As stated by Sinclair and Smith [69], security tends to ignore such "real-world subtleties".

Real-world subtleties encompass social dimensions of socio-technical systems, such as the usability [5] and conviviality [40] of the system. These human factors should be taken into account from the early phases of the development of socio-technical systems. Several lines of research (e.g., [2, 11, 32, 33, 38, 61]) have explored the problem of designing socio-technical systems aiming to achieve trade-offs between usability and security. However, there are other social dimensions of socio-technical systems which may conflict with security. In this paper, we study the trade-off between security and conviviality. Conviviality is a concept from the social sciences defined by Illich as "individual freedom realized in personal interdependence" [40]. Clearly, a tension exists between conviviality and security: too much security threatens conviviality, while being convivial is a potential threat to confidentiality and privacy. Such trade-off/potential conflicts should be identified and managed as soon as possible, at requirement level and early design stages. Producing a security policy model that is non-conflictual with conviviality expectations is desirable, since it encourages actors to respect the security policy being used.

These initial considerations raise the main research question addressed in this paper: *How to manage the two different orthogonal concerns of security and conviviality in the elaboration of system operational requirements and design models?*

The main contribution of the paper is in (1) bridging the gap between two concepts that are rarely brought together, i.e., security and conviviality, which are often considered opposite domains of Social Sciences and Informatics, by defining a socio-technical mapping that bridges the gap between conviviality and access control; (2) handling access control policy update on the basis of conviviality recommendations.

The paper studies how to enable a symbiotic elaboration of a security policy together with a conviviality model, so that the potential conflicts between these two viewpoints can be detected and solved. Starting from initial operational requirements, the approach makes the trade-off explicit, and results in a representation of convivial-

ity consistent with the security policy (that can be updated during the process). The approach does not intend to weave or compose these viewpoints [58,56]; rather, it produces two consistent models, one for security and one for conviviality. On the one hand, the security model, in the hand of security officers, describes the security policy to be enforced; on the other hand, the conviviality model, in the hand of business experts, is a formal tool for reasoning and improving this social dimension of a system.

To recombine these two viewpoints, we propose the DN-AC alignment methodology which allows the specification of access control policies compatible with a conviviality-driven specification of a system. Increasing conviviality while keeping a system secure raises the question of how to adapt an access control policy while increasing the conviviality of the system. Thus, our main research question breaks down into the following subquestions:

1. How to model access control policies to make explicit its adaptable parts?
2. How to model conviviality from an initial use case scenario?
3. How to update and adapt access control according to conviviality improvements?

The DN-AC methodology uses the SI* modeling language [36,50] to capture the socio-technical requirements of a system, in particular security and conviviality requirements, and to generate the access control policy from requirements models [51]. To address question (1), we differentiate non-negotiable authorizations from negotiable authorizations: negotiable authorizations correspond to the adaptable parts of the policy, whereas non-negotiable authorizations correspond to socio-technical requirements that must be implemented in the system. To address question (2), abstracting from method-specific concepts such as plans and actions, we create a dependence network from the SI* model. Dependence networks [23,67] offer an efficient model for representing interdependencies among agents. Dependence networks are labeled directed graphs where the nodes are agents, and each labeled edge represents that the former agent depends on the latter one to achieve some goal. The focus on dependence networks and more specifically on their cycles, is a reasonable way of formalizing conviviality as something related to the freedom of choice of individuals together with the subsidiary relations – interdependence for task achievement – among fellow members of a social system. Intuitively, more opportunity to work with other people increases the conviviality in the system. The analysis of the dependence network could lead to an increase in the number of potential coalitions in the dependence network, and hence increase the conviviality of the system. This process allows us to address question (3).

Our motivation is to address the challenge of finding trade-offs between access control policy and conviviality, particularly as they pertain to the Ambient Assistant Living (AAL) domain emphasized by our partner, Luxembourg HotCity. We thus aim to provide guidelines to assist system designers to improve many aspects of daily life of Luxembourg citizens which are illustrated through our different case studies. Nonetheless, our approach is general and can be applied to other application domains in which conviviality is an important requirement.

The structure of this paper is as follows. First, we introduce our methodology in Section 2 and our use cases in Section 3. We then present the building blocks of the methodology: requirements modeling with SI* in Section 4, access control policy in Section 5, and dependence networks in Section 6. In Section 7, we propose a mapping between access control policies and dependence networks along with the policy updating/adapting process. In Section 8, we validate our approach through the scenarios of the Ambient Assisted Living (AAL) from the HotCity of Luxembourg. Finally, we dis-

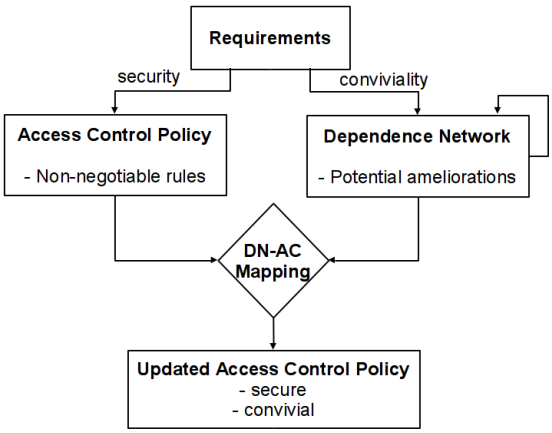

Fig. 1 DN-AC Alignment Methodology

cuss related work in Section 9, and conclude the paper providing directions for future work in Section 10.

2 DN-AC Alignment Methodology

This section presents a methodological framework for generating conviviality-driven access control policies from socio-technical requirements. The aim of the methodology is to find a trade-off between security, more specifically access control constraints, and conviviality. This trade-off makes it possible to consider new dependencies between agents, while the system remains secure. Figure 1 illustrates the overall process.

First, the security and functional requirements of the system are elicited and represented in a requirements model using a goal-oriented framework [36, 50]. The requirement model captures stakeholders and system actors together with their goals, the tasks to achieve those goals, required resources, and the security and functional dependencies among them. The requirements model is formally analyzed against a number of security properties to verify whether requirements have been specified correctly. If the model satisfies these properties, it is used to determine the authorization rules which are necessary to protect the system (left side of Figure 1) and to analyze the conviviality of the system (right side of Figure 1).

An *access control policy* is a set of authorization rules that specify the conditions under which users are authorized/denied to access the protected data or resources. Positive authorization rules refer to permissions to access resources, while negative authorization rules refer to prohibitions to access resources. In this paper, we distinguish between *negotiable* and *non-negotiable* authorization rules. Non-negotiable authorization rules correspond to hard requirements, i.e. requirements that must be fulfilled to guarantee the security of the system and, therefore, they cannot be modified; on the other hand, negotiable authorization rules correspond to the adaptable part of the access control policy and can be modified, for instance, to increase the conviviality of the system. For the specification of the access control policy, we use a model-driven approach based on [51]. In particular, the access control policy is derived by analyzing the duties and responsibilities assigned to stakeholders and system actors.

In parallel and independently from the specification of access control policy, *dependence networks* are created from the requirements model and used to reason about the potential ameliorations to increase the global conviviality of the system. Different techniques for improving conviviality have been proposed in the literature [9, 15], for example by changing the agents within the system, by changing the dependencies among them, by introducing or changing normative dependencies, and by changing the composition of the coalitions. In this paper, we increase the number of coalitions between agents by adding/removing dependencies between agents.

The analysis of the dependence network may suggest some potential dependencies between agents to increase the conviviality of the system. Furthermore, feedback gathered from users, for example through comments or direct input, may be used as additional information to represent users dependencies among each other in the dependence networks. Such dependencies may also impact resource sharing between agents, and thus the authorization rules should be updated accordingly.

The final step of the process aims to reconcile the security and conviviality viewpoints in order to generate "secure and convivial" system. This step involves a socio-technical mapping between the access control policy and the dependence network (referred to as DN-AC mapping in Figure 1). The mapping relates each goal of the dependence network to the corresponding authorization rules. If the authorization rules are negotiable, they may be changed in order to increase the system conviviality; otherwise, if the authorization rules are non-negotiable, the corresponding ameliorations are discarded. Indeed, the revised policy should not violate the security requirements of the system.

3 Use Case Scenario

We have considered 12 use case scenarios that have been elaborated and validated together with the HotCity Ambient Assistant Living (AAL) of Luxembourg. The scenarios illustrate how a Home Care System (HCS) could improve its users' quality of life in a variety of cases and cover different areas and problems related to AAL like health problems (Heart-attack, Fever, Medication, Alzheimer), psychological or social problems (Loneliness, Isolation, Depression, Alcoholism) and economical problems (Finances). A complete description and analysis of these scenarios can be found in [71] where the scenarios are represented graphically using dependence networks (DN). The access control policies that could be applied to these scenarios to guarantee the security of the system are also given in [71]. These scenarios are summarized in Table 4.

In this paper, we have selected the scenario entitled 'Heart-attack 1' from the use cases (Table 4) as our running example:

Ms. Annette Becker is eighty-five years old. She is prone to heart failures; hence the hospital has installed a smart Home Care System (HCS) at her house. Suddenly, as she walks out of the kitchen, she stumbles, falls down and cannot get up. In real time, her health bracelet sends information, such that heart beat and skin temperature, to the HCS. The system analyzes the images captured on the video monitoring system and gets an updated medical profile of the patient from the hospital. By combining all the different pieces of information, the system infers that it is a medium emergency situation. In such a case, the emergency calls list specifies Annette's neighbors as primary contact. The HCS (via a phone communication system) sends the neighbors an

ID	Scenario Title	Scenario Description
1	Heart-attack 1	HCS monitors a patient prone to heart failures and provide her with social support
2	Loneliness	HCS arranges a birthday party for a lonely senior citizen
3	Isolation	Sending reminders to family members to call their elderly relatives during occasions
4	Finances	Financial support from the family and legal support from an expert, arranged by HCS
5	Fever	A patient with fever does not feel helpless, after talking to his doctor and receiving the right medication
6	Medication	HCS monitors patient to take her medication, and take action if she does not
7	Weight	Significant weight gain, recognized and solved by HCS
8	Depression 1	Depression, expressed through inactivity, is surpassed with the help of WAS
9	Alzheimer	Alzheimer patient finds his way home thanks to his GPS/video capture/HCS
10	Depression 2	HCS records lower activity level and contacts a neighbor to visit senior citizen
11	Alcoholism	Alcoholism prevented by the HCS with the help of the community
12	Heart-attack 2	HCS captures patient's heart attack danger and infers to contact family for help

Table 1 Use Case Scenarios

SMS inquiring whether they are available to come and help; if no neighbor is available, a friend or family member is contacted.

For Annette, being helped by people she feels connected to is important. To this end, she is also given access to social support, such as social assistance and support groups, that she can use to get health-related information and to which she may contribute. However, the system should guarantee that no private information such as video captures or medical data falls in mischievous hands. Therefore, the system should be convivial but stay private.

4 Requirements Modeling

To capture stakeholders' requirements, we adopt the SI* modeling language [36,50] as it supports the analysis of security and dependability requirements of socio-technical systems. SI* extends the i* framework [73] and is founded upon the concepts of *actor*, *goal*, *resource*, and *social relations*. An actor is an active entity that has strategic goals. A goal is a state of affairs whose realization is desired by some actor (objective), can be realized by some (possibly different) actor (capability), or should be authorized by some (possibly different) actor (entitlement). Entitlements, capabilities and objectives of actors are modeled through relations *own* (**O**), *provide* (**P**) and *request* (**R**), respectively. Resources are artifacts produced/consumed by goals. Goals can be refined using *AND/OR decomposition*; *means-end* relations identify resources produced or consumed by a goal.

Social relations are captured by the notions of *trust* and *delegation*. Trust is a relation between two actors representing the expectation of one actor (*trustor*) about the behavior of the other actor (*trustee*). In particular, *trust of execution* (**Te**) models the trustor's expectations about the ability and dependability of the trustee in achieving a goal; and *trust of permission* (**Tp**) models the trustor's expectations that the trustee

does not misuse the given permission. Delegation is used to represent a formal passage of responsibility (*delegation of execution* (**De**)) or authority (*delegation of permission*) between two actors (*delegator* and *delegatee*).

As in [3], we distinguish three types of permission: *access*, *modify*, and *manage*. Each type of permission determines the set of actions that an actor can perform. For instance, if an actor has *access* permission on a resource, he can only use the resource. *Modify* permission allows an actor to change the content of a resource. *Manage* permission allows an actor delegate the permission to other actors and modify permissions of other actors. The three types of permission, *access*, *modify* and *manage*, operate in a hierarchical manner. In particular, *manage* permission implies *modify* permission, and *modify* permission implies *access* permission. Delegation of access/modify/manage is graphically represented by an edge with label **Dpa**, **Dpmd** and **Dpma**, respectively.

In socio-technical systems, an actor who has the right to achieve a goal, may delegate such rights to the wrong actor. Since many actors may have the right to achieve the goal, it is not always possible to control that an actor cannot achieve a goal. To address this issue, we model explicit negative authorizations as in [37]. A negative authorization expresses a *denial* for an actor to achieve a goal or access a resource. As for positive authorizations, we distinguish three types of denial. Their meaning is dual to the one of positive authorizations (the hierarchy of types of permission is also reversed). Denial of access/modify/manage is graphically represented by an edge with label **Dla**, **Dlmd** and **Dlma**, respectively. In our approach, negative authorizations take precedence over positive ones, i.e. whenever a user has both a positive and a negative authorization on the same goal/resource, the user is prevented to access it. Moreover, an actor can deny permission to another actor only if he has *manage* permission.

Figure 2 presents the SI* model corresponding to the AAL scenario presented in Section 3. The scenario involves five actors: *Patient*, *Hospital*, *HCS*, *Neighbor*, and *Social Support*. The *Patient* has the intention (objective) to fulfill goal *stay healthy* and is the legitimate owner of her data, while the *Hospital* has the capability to achieve goal *update patient record*. Goals can produce or consume resources. For instance, goal *maintain patient record* requires resource *patient profile*. The *Patient* relies on the *HCS* to fulfill goal *stay healthy*. In turn, this goal is decomposed into *update patient profile*, *determine emergency level*, and *provide medical support*. In order to achieve these goals, the *Patient* grants access to her data to the *HCS* (through the *Hospital*). For privacy reasons, the *Patient* denies access to her data to the *Neighbor*.

5 Access Control Policy

Access control is a security mechanism which is typically adopted for the protection of sensitive information. An access control policy consists of rules that specify which actions a user can or cannot perform on system resources. Hereafter, we assume that there is a centralized access control system that enforces access control policies.

Definition 1 Let S be the set of subjects, R the set of system resources, A the set of actions, and $\Pi = \{permit, deny\}$ the set of rulings. An *authorization rule* is a 4-tuple $\langle s, a, r, \tau \rangle$ where $s \in S$, $a \in A$, $r \in R$, $\tau \in \Pi$. A positive authorization rule $\langle s, a, r, permit \rangle$ specifies that a subject s is allowed to perform action a on resource r, whereas a negative authorization rule $\langle s, a, r, deny \rangle$ states that a subject s is prohibited to perform action a on resource r. An *access control policy* is a set of (positive and negative) authorization rules.

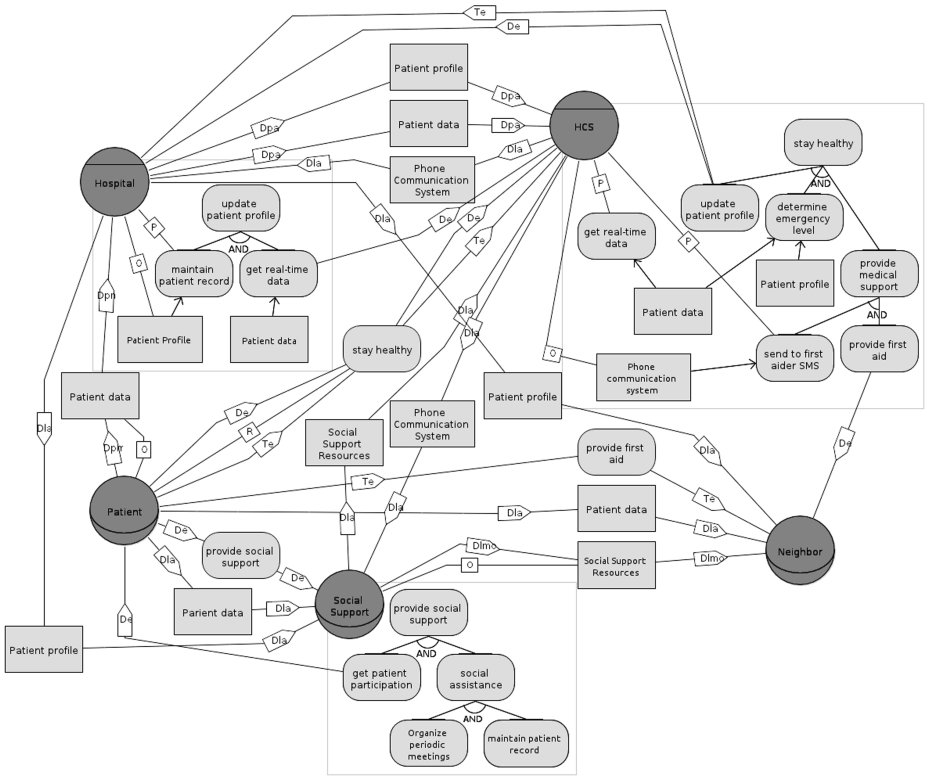

Fig. 2 SI* Model of the HotCity Ambient Assistant Living (AAL) scenario

We argue that one desideratum for the specification of access control policies is that such policies are derived (possibly automatically) from the requirements of the system in which they will be eventually enforced. This ensures the alignment of the policy specification with the requirements and reduces the efforts of policy writers making their tasks less error-prone. To this end, we adopt an approach similar to the one proposed in [51]. First, the requirements model is analyzed using the tool in [48] to determine whether the designed system satisfies a number of authorization, availability and privacy properties. Such properties aim, for instance, to verify whether actors have the necessary permissions to perform their assigned duties, and whether the elicited permissions comply with the need-to-know principle. This guarantees that on the one hand, the elicited authorizations are sufficient for a complete business execution, and thus that the obtained access control policy is not too restrictive; on the other hand, it also guarantees that actors does not have permissions they do not need. Once the requirements model satisfies these security properties, the security elements of the model (i.e., ownership and delegation of permission) are used to define access control policies.

In SI*, own denotes the entitlements of actors: an actor (the owner) has full authority concerning his resources. Accordingly, we assume that the owner of a resource

Positive Authorizations	
1	$\langle Patient, manage, patient\ data, permit\rangle$
2	$\langle Hospital, manage, patient\ profile, permit\rangle$
3	$\langle Hospital, manage, patient\ data, permit\rangle$
4	$\langle HCS, manage, phone\ communication\ system, permit\rangle$
5	$\langle HCS, access, patient\ data, permit\rangle$
6	$\langle HCS, access, patient\ profile, permit\rangle$
7	$\langle Social\ Support, manage, social\ support\ resources, permit\rangle$
Negative Authorizations	
8	$\langle Hospital, access, phone\ communication\ system, deny\rangle$
9	$\langle HCS, access, social\ support\ resources, deny\rangle$
10	$\langle Social\ Support, access, patient\ data, deny\rangle$
11	$\langle Social\ Support, access, patient\ profile, deny\rangle$
12	$\langle Social\ Support, access, phone\ communication\ system, deny\rangle$
13	$\langle Neighbor, access, patient\ data, deny\rangle$
14	$\langle Neighbor, access, patient\ profile, deny\rangle$
15	$\langle Neighbor, modify, social\ support\ resources, deny\rangle$

Table 2 Non Negotiable Authorizations for the AAL scenario

has the highest permission on the resource (i.e., *manage* permission). Remark that high permissions imply lower permissions. Entitlements of actors are then propagated through delegation of permission. In particular, an actor can delegate any type of rights on a resource to another actor if he has *manage* rights. This propagation is enabled through axioms that permit to derive authorizations from the facts obtained through the transformation of graphical models (see [51] for details). Intuitively, authorizations of a given actor are derived by determining the existence of a chain of delegations of permission from the owner to that actor (possibly through goal refinement). Negative authorizations are derived similarly by determining if the actor denying another actor to use a given resource has legitimately received *manage* rights from the resource owner.

Table 2 presents the access control policy derived from the SI* model in Figure 2. One can observe in the table that several authorizations (either positive or negative) are not specified (e.g., no authorization on the phone communication system are defined for the patient). Indeed, unlikely elicited requirements cover all possible cases. We assume that the authorization decision for cases where an authorization rule is not defined, is deny. Although this solution guarantees that the deployed access control mechanism complies with the least privilege principle, it has the side effect that such a mechanism may be too restrictive.

To address this issue, we distinguish between *negotiable* and *non-negotiable* authorization rules. Non-negotiable authorization rules represent rigid authorizations that cannot be modified to guarantee the security of the system. Essentially, they are strict constraints imposed by the requirements and, therefore, they cannot be relaxed. Accordingly, the authorizations in Table 2 which are translated from requirements are non-negotiable. Negotiable authorization rules, on the other hand, regulate situations for which a constraint is not explicitly defined by the requirements. These rules are derived from conviviality recommendations (Section 7). Intuitively, the distinction between non-negotiable and negotiable authorization rules resembles the distinction between hard requirements (i.e., compulsory requirements) and soft requirements (i.e., optional requirements) [41].

By introducing negotiable authorization rules, we aim to increase the flexibility of the system by highlighting the adaptable part of the authorization policy, rather than introducing other decisions types. In other words, unlike XACML [1] which extends binary decisions *Permit* and *Deny* with *Not Applicable* decision to indicate that no policies are applicable to a given access request, we assume that the default access decision is deny and provide the flexibility necessary to deal with requirements evolution through negotiable authorizations.

6 Conviviality Model

Conviviality has recently been introduced into multi-agent and ambient intelligence systems [18,21] to highlight soft qualitative requirements like the user friendliness of systems. The concept of conviviality, originated from social science, was popularized by a book of Illich in 1973 called "tools for conviviality", in which he defines conviviality as:

"individual freedom realized in personal interdependence" [40]

Interdependence and dependencies play a prominent role in many formal systems such as, for example, Bayesian networks. In this paper, the notion of dependence is used as it is in multi-agent systems where dependency relations relate agents who seek to reach their goals, to other agents who have the abilities required to fulfill these goals. Following conventions in game theory as well as multi-agent systems, we say that the ability of an agent to fulfill goals of other agents is an indication of its social power.

A dependence network is a social network where the relations among the agents are labeled with a goal, expressing that an agent depends on another agent for the fulfillment of this goal. Based on the notion of social dependency introduced in [23], dependence networks were put forward in [66], and further developed in [67]. Boella et al. [8] show how dependence networks can be used to determine which reciprocity-based coalitions can be formed. In [21], these coalitions represent potential interactions among the agents, and are thus an indication of conviviality.

Conviviality can be measured by the number of reciprocity-based coalitions that can be formed, as developed in [16]. Some coalitions, however, provide more opportunities for their participants to cooperate with each other than other coalitions, being thereby more convivial. For example, if a patient needs assistance from her neighbors, it is better if she can rely on several neighbors and is able to choose among them the one(s) with whom she already has interactions. The relation is reciprocal as they both give to the coalition and receive from the coalition; it is more convivial.

6.1 Conviviality in Access Control

In the SI* modeling language [36,50], actors are endowed with intentionality from the early phases of the system development process. This allows for a profound understanding of the environment and of the interactions between stakeholders.

SI* uses the concepts of entitlement, objective, and capability to distinguish between actors who want access to a resource or the fulfillment of a goal, from actors who have the capabilities to provide a resource or fulfill a goal, and actors who are entitled

to do any of the above. Entitlements and their delegation are the basis for the specification of the access control policies of the system. In particular, the entitlements of actors are mapped into positive and negative authorization rules which specify whether an actor is allowed or denied to access data and IT resources, and thus to fulfill its goal.

Stakeholders may not have the capabilities to fulfill their objectives. The social relation of delegation is used in SI* to represent a formal passage of responsibility between two actors, namely the *delegator* and the *delegatee*. More specifically, SI* uses *delegation of execution* to represent that an agent delegates to another agent the fulfillment of an objective, thereby creating a dependency between the two agents in relation to the objective. For example, in the SI* model of the HotCity Ambient Assistant Living scenario illustrated in Figure 2, the patient delegates to the HCS the execution of its goal *stay healthy*, which, in effect, makes the patient dependent on the HCS for that very same goal. We therefore use the concept of delegation of execution to build a bridge between conviviality and access control.

6.2 Dependence Networks

We now introduce our definition of dependence networks. Note that in our model, the dependencies are among the agents, so if an agent a depends on a distinct agent b for an action, a resource or a plan to achieve its goal g, the dependency of agent a towards agent b will be on g. Goals are considered the *reasons* for which the dependencies arise. Abstracting away from the actions, resources and plans of the agents, we define a dependence network as follows:

Definition 2 A *dependence network* is a tuple $\langle A, G, dep, \geq \rangle$ where: A is a set of agents, G is a set of goals, $dep : A \times A \rightarrow 2^G$ is a function that relates with each pair of agents, the sets of goals on which the first agent depends on the second, and $\geq$: $A \rightarrow 2^G \times 2^G$ is for each agent a total pre-order on sets of goals occurring in his dependencies: $G_1 \geq (a) G_2$.

In our model, the dependencies in the dependence network correspond to the delegations of execution in the SI* model, which account for goal refinement as some delegated goals are subgoals of other delegated goals.

Returning to our scenario, consider the dependence network $DN = \langle A, G, dep, \geq \rangle$ corresponding to the SI* model introduced in Section 4:

- Agents $A = \{P, H, HCS, N, S\}$, respectively: patient, hospital, HCS, neighbor, and social support; Goals $G = \{g_1, g_2, g_3, g_4, g_5, g_6, g_7\}$.
- $dep(P, HCS) = \{g_1\}$: agent P depends on agent HCS to achieve goals $\{g_1\}$, *stay healthy*;
  $dep(HCS, H) = \{g_2\}$: agent HCS depends on agent H to achieve goals $\{g_2\}$, *update patient profile*;
  $dep(H, HCS) = \{g_3\}$: agent H depends on agent HCS to achieve goals $\{g_3\}$, *get real-time data*;
  $dep(HCS, N) = \{g_4\}$: agent HCS depends on agent N to achieve goals $\{g_4\}$, *provide first aid*;
  $dep(P, S) = \{g_5\}$: agent P depends on agent S to achieve its goals $\{g_5\}$, *provide social support*;

$dep(S, P) = \{g_6\}$: agent S depends on agent P to achieve its goals $\{g_6\}$, *get patient participation.*
- Agent P prefers to stay healthy than to get social support: $\{g_5\} >_{(P)} \{g_6\}$

The graphical representation of the dependence network is illustrated in Figure 3. The figure should be read as follows: the five agents are represented by the nodes in the graph, and the dependencies among them are indicated by labeled arrows. The label indicates the goal on which the dependency is based. For example, the patient depends on its Home Care System to stay healthy.

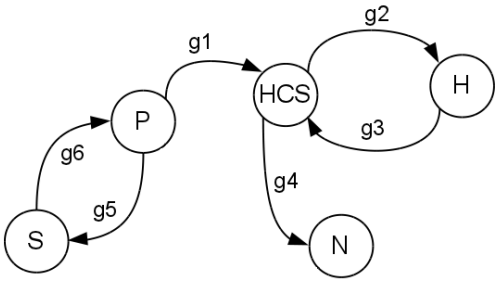

Fig. 3 Dependence Network DN

In socio-technical systems, agents are involved with each other and may support each others' goals if an agent is not able to achieve them by itself. Dependence networks and coalitional game theories can be used to define potential reciprocity-based coalitions, which are sets of agents together with a subset of the dependencies for these agents, such that each agent contributes something and receives something from the coalition. Based on [9], we define a coalition as follows:

Definition 3 Let A be a set of agents and G be a set of goals. A *coalition function* is a partial function $C : A \times 2^A \times 2^G$ such that $\{a \mid C(a, B, G)\} = \{b \mid b \in B, C(a, B, G)\}$, the set of agents profiting from the coalition is the set of agents contributing to it. Let $\langle A, G, dep, \geq \rangle$ be a dependence network, a coalition function C is a *coalition* if $\exists a \in A, B \subseteq A, G' \subseteq G$ such that $C(a, B, G')$ implies $G' \in dep(a, B)$.

The priority relation can be taken into account to define preferred reciprocity based coalitions. Like conviviality, coalitions emerge from the sharing of properties and/or behaviors whereby each member's perception is that their personal needs are taken care of.

In order to evaluate the conviviality in the network, we first make the following assumptions (or hypotheses):

H1 The cycles identified in a dependence network are considered as coalitions.[1] These coalitions are used to evaluate cooperation in the network. Cycles are the smallest graph topology expressing interdependence, thereby cooperation, and are therefore considered atomic relations of interdependence. When referring to *cycles*, we are implicitly signifying *simple cycles*, i.e., where all nodes are distinct [24]; we also discard self-loops. Moreover, when referring to cooperation, we always refer to potential interaction rather than to actual interaction.

H2 There is more conviviality in larger coalitions than in smaller ones. We express this hypothesis through the following two cases. First case, a dependence network DN_i with a coalition of size n is better for conviviality than a DN_j with coalition of size $m < n$. For example, consider a coalition for peace in the world. The more countries participate, the better it is. Second case, a dependence network DN_i with a coalition of size n is better for conviviality than a dependence network DN_j with two coalitions, one of size k and the other of size l, such as that $k + l \leq n$, all else being equal. This is motivated by the fact that having one large coalition eliminates the risk of being exposed to potential competition from other coalitions, which may be looking for the same resources.

H3 The more coalitions in the dependence network, the higher the conviviality measure (ceteris paribus).

Our aim is to maximize cooperation in the system. Thus, our requirements are:

R1 maximize the size of the agent's coalitions by increasing the number of agents involved in the coalitions,

R2 maximize the number of these coalitions.

Intuitively, the goal is hence not only to have as many agents taking part in the largest coalition(s), but also have as many coalitions among the participating agents.

Dependence cycles in the network indicate potential interactions and coalitions among the agents. Thus, we analyze cycles and their configurations in the network. The dependence network in Figure 3 contains two cycles which are indicative of two potential coalitions, on the one hand among agents HCS and H, and on the other hand among agents P and S. We indicate the two potential coalitions as follows: $C_1 : \{(H, HCS, g_3), (HCS, H,\ g_2)\}$ and $C_2 : \{(P, S, g_5), (S,\ P,\ g_6)\}$, where we write (a, b, g_1) for $(a, b, \{g_1\})$.

Note that agent N does not depend on any other agent, whereas agent HCS depends on agent N for goal g_4. Hence, agent N has no incentive to satisfy agent HCS's goal, as it does not have any goal to reciprocate. This indicates that there may be ways to increase the conviviality of the network, for example by including into a coalition agents, such as agent N, which are not part of the coalition.

[1] Note that the terms "cycle" and "coalition" represent two distinct realities. Keeping the terms different is consistent to the domains they belong to: a coalition describes a set of agents and comes from agents domains and game theory, while cycle is a graph-theoretical term. The dependence relations among agents participating to a coalition can be analyzed in terms of coalitions, not cycles – which would not mean anything. Furthermore, we count the cycles in the graph; counting coalitions would be inexact, as such a term does not exist in graph theory. Nonetheless, there exists a relation between two terms: cycles identified in a dependence network are considered as coalitions.

6.3 Conviviality Increase

According to Boella et al. [9], coalitions in a dependence network may be changed in the following ways: 1) by changing the agents, e.g., by entering or leaving the system, 2) by changing the dependencies among the agents, i.e., by adding or deleting dependencies among the agents, 3) by introducing or changing normative dependencies, such as obligations and prohibitions, and 4) by changing the composition of the coalitions while the agents and dependencies remain the same.

In this paper, we assume that the set of agents within the dependence network is given and it does not change over time. Similarly, we do not consider changes in the composition of the coalitions within the network due to internal processes. Finally, we do not introduce normative dependencies as, typically, policies are considered as rules and constraints that model intended behaviors. In fact, they contrast with norms considered as *agreed policies* in the sense that they are agreed to by the members of a community. Conviviality for example, is usually considered as a social norm. Norms apply to groups and regulate the behavior of the individuals among themselves; they differ from policies, such as access control policies, which may also apply to single individuals. For example, privacy policies may apply to an individual patient, and mail filtering policies to a single doctor. Thus, among the approaches mentioned in [9], we adopt the second approach; accordingly, a change in the network is only due to the change of a dependency between two agents.

We recall from the previous section, the two requirements for conviviality, i.e., to maximize the number of agents involved in coalitions ($R1$) and the number of coalitions ($R2$). Satisfying $R1$ and $R2$ will maximize conviviality.

Consider that a need for social interactions may be inferred for the neighbor, or directly expressed by the neighbor through a feedback loop. Such an aspiration could be fulfilled based on distinct dependencies, i.e, agent N (i.e., the neighbor) may depend on agent P (i.e., the patient), or on agent S (i.e., Social Support) to achieve it. As the access to the social support of the patient is managed by the Social Support agent itself, it is thus the Social Support agent that would have to create the dependency. We now present the mechanism that would allow such dynamics.

6.4 Dynamic Dependence Networks

The notion of agents' power introduced in [23] allows an agent to add dependencies inside a network, i.e., to increase reciprocity-based coalitions, and thus conviviality. Dynamic dependence networks proposed in [21] allow the possibility to introduce new dependencies. We define a dynamic dependence network DDN as in [21]:

Definition 4 A *dynamic dependence network* is a tuple $\langle A, G, \mathit{dyndep}, \geq \rangle$ where:

- A is a set of agents and G is a set of goals.
- $\mathit{dyndep} : A \times A \times A \rightarrow 2^G$ is a function that relates with each triple of agents the set of goals on which the first agent depends on the second, if the third agent creates the dependency.
- $\geq$: $A \rightarrow 2^G \times 2^G$ is for each agent a total pre-order on goals which occur in his dependencies: $G_1 \geq(a)\ G_2$.

Building on the dependence network DN defined in Section 6.2, consider the following dynamic dependence network $DDN = \langle A, G, \mathit{dyndep}, \geq \rangle$ where:

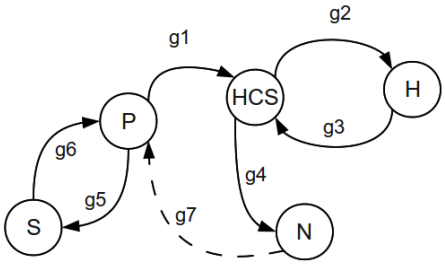

Fig. 4 Dynamic Dependence Network DDN with the added goal g_7, in dashed line

1. $A = \{P, H, HCS, N, S\}$; $G = \{g_1, g_2, g_3, g_4, g_5, g_6, g_7\}$, now including g_7: *get social interaction*.
2. Added dependency: $dyndep(N, P, S) = \{g_7\}$: agent N depends on agent P to achieve goal $\{g_7\}$ if it is created by agent S.[2]
3. Preferences on goals are as previously.

Figure 4 illustrates dynamic dependence networks DDN. The dashed arrow labeled with goal g_7 represents the new dependency in DDN. The added dependency creates a new cycle in the network, indicating an additional potential coalitions, namely C_3 : $\{(N, P, g_7), (P, HCS, g_1), (HCS, N, g_4)\}$. In this coalition, the three agents N, P and HCS depend on each other to achieve their goals; agent N is no longer isolated. The new coalition C_3 ensures that the agent N satisfies goal g_3 of agent HCS. The dynamic dependence network DDN is more convivial than the dependence network DN as there are more cycles in the graph, indicating more potential coalitions among the agents.

More dependencies may be considered to create more cycles in the network, thereby increasing the potential interactions among the agents, and therefore the conviviality of the system. For instance, coalition C_4 : $\{(P, HCS, g_1)$, $(HCS,\ N,\ g_4)$, (N, S, g_7), $(S, P, g_6)\}$ could be created for instance by triggering $dyndep(N,\ S,\ S) = g_7$. However, no such a dependency has yet been elicited, and therefore this coalition cannot be considered at this point.

6.5 Discussion

There is a large amount of work on how to use dependence networks, specially since we are interested in the resolution of conflicts and regulation. Dependence networks, firstly defined by Emerson [31], have been developed in the context of multi-agent systems by Conte and Sichman [67]. Sichman [66] presents coalition formation using a dependence-based approach where a *dependence situation* allows an agent to evaluate the susceptibility of other agents to adopt his goals. Sauro [70,62] shows how to use dependence networks to discriminate among different potential coalitions during the coalition formation process. He assumes that a coalition is effectively formed only when

[2] Note that a dependency *dep* in DN can be seen as a particular case of dependency *dyndep* in DDN.

all its members agree on it and they cannot deviate from what was established in the agreement, once they decide to enter it. Bonzon et al. [10] use dependence networks to compute pure-strategy Nash equilibrium in a simpler way, without enumerating all combinations of strategies. The notion of dependence between players and variables is used to split up a game into a set of interacting smaller games, which can be solved more or less independently. In Sauro and Villata [63], abstract and refined dependence networks for cooperative boolean games are introduced to improve the computation of the core. Koller and Milch [43] introduce a representation language for multi-player games called multi-agent influence diagrams. It extends the graphical models developed for probability distributions to a multi-agent decision-making context. Like in dependence networks, these diagrams explicitly encode a structure involving the dependency relationships among variables.

Many examples of using dependence networks can be found in software engineering. For instance, the i* modeling language [73] and the Tropos software engineering methodology [12] represent the network of dependency relationships among the actors to analyze the organizational setting in which the system-to-be operates. In particular, their notation allows the description of the *structural* aspects of the early requirements model, in terms of relationships and dependencies among actors. These frameworks have been extended to describe also how the network of dependencies evolves over time and the circumstances under which a given dependency arises and can be specified, as well as the conditions that permit to consider the dependence to be fulfilled [35, 52].

One of the main advantages of dependence networks is that they can be rewritten as power structures: a (social) dependency of agent d on agent p for reason e can be conceptualized as the (social) power of agent p over agent d for the reason e. Moreover, the distinction between *reciprocal* and *mutual* dependencies [68] involves the development of a social reasoning mechanism that analyzes the possibilities to differently profit from reciprocal than from mutual dependencies.

Efficiency and stability metrics are commonly used to evaluate coalitions. The former giving an assurance on the economical gain reached by being in the coalition, the later giving a certainty that the coalition is viable on the long term. Therefore, the positive evaluation of a coalition against these two metrics is often considered to be a prerequisite for the coalition formation. However, depending on the application domain, other functional and non-functional requirements, e.g., security, user-friendliness or conviviality, may play an important role in the choice of a coalition. Requirements may be considered in a trade-off at the same level as efficiency and stability, or as a further filtering criterion, to select among otherwise efficient and stable coalitions.

We do not introduce temporality in our cooperation measures as they aim to assess the conviviality of the system at design time. In particular, cooperation measures quantify interdependence in social relations, representing the degree to which the system facilitates social interactions. Intuitively, more interdependence increases cooperation among groups of agents or coalitions, whereas larger coalitions may decrease the efficiency or stability of these involved coalitions. In contrast, run-time evaluation would require the use of temporal dependence networks as the ones proposed in [17, 20] which analyze the evolution of cooperation over time.

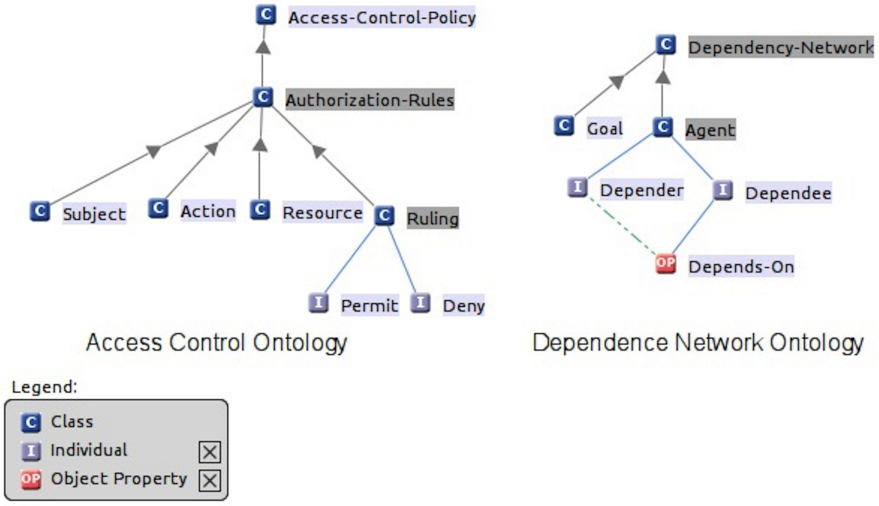

Fig. 5 Dependence Network and Access Control Ontologies

7 Access Control Policy Update

The analysis of the dependence network may lead to consider some potential coalitions and mutual dependencies between agents in order to increase the conviviality of the system. Those potential dependencies may require granting additional authorizations to agents. This, however, may compromise the security of the system. Therefore, we need to analyze the impact of potential dependencies on the existing access control policy. This section describes how potential dependencies captured through the analysis of the dependence network are used to adapt the access control policy regulating the overall system. First, we discuss the semantic gap between the ontology related to access control and the ontology related to dependence networks. Then, we present our approach to bridge such a gap. Finally, we present a strategy for policy update based on the distinction between negotiable and non-negotiable authorization rules.

7.1 Semantic Gap between Access Control Paradigm and Conviviality Paradigm

We use ontologies to represent access control concepts as well as dependence network concepts. An ontology defines a formal representation of the concepts and relationships between those concepts in a particular domain [65]. Figure 5 illustrates the ontology related to access control (left) and the one related to dependence networks (right), both visualized using NeOn toolkit [54]. In the access control ontology, authorization rule is modeled as a subclass of class access control policy; classes subject, action, resource, and ruling are modeled as subclasses of class authorization rule. Rulings permit or deny are modeled as individuals (i.e., instances). In the dependence network ontology, classes goal and agent are subclasses of class dependence network; the depender and dependee are modeled as individuals which are related by object property depends on.

As shown in Figure 5, access control is shaped by the notions of subject, resource, action and ruling. Those concepts, however, do not appear in the dependence network ontology. Therefore, adapting an access control policy on the basis of dependency relations between agents requires closing the semantic gap between these two ontologies. In the remainder of the section, we discuss how to create a mapping between dependence network concepts and access control concepts to narrow the semantic gap between the two paradigms.

7.2 Mapping Between Access Control Policy and Dependence Networks

Potential dependencies are built upon the achievement of a specific goal between a depender and a dependee. To be able to analyze the impact of such dependencies on the existing access control policy, it is necessary to determine which authorization rules an agent needs in order to carry out the assigned duties (i.e., to fulfill the delegated goal). To bridge the gap between dependence networks and access control, we propose to map each goal in the dependence network to the set of actions and resources that are required to fulfill the goal. This mapping is illustrated by a socio-technical mapping matrix defined as follows.

Definition 5 A *socio-technical mapping matrix* is a $n \times m$ matrix where rows denote pairs $(resource, action)$, and columns denote goals.

The socio-technical mapping matrix shows, for each goal in the dependence network, which resources are needed for the achievement of the goal together with the actions (i.e., access, modify and manage) that can be performed on such resources. The link between resource and goals is derived from the SI* model through AND/OR refinement and delegations of execution (Section 2) using the approach presented in [51]. Intuitively, if a resource is linked to the goal (via a means-end relation), then the resource is needed for the achievement of the goal. If a goal is decomposed into subgoals, each subgoal is iteratively analyzed. In particular, resources linked to a subgoal are needed for the achievement of the upper level goals.[3] If a (sub)goal is delegated to another actors, the corresponding goal model rooted in the rationale of the delegator is analyzed as described above. Thus, the set of resources needed to achieve a goal includes all resources needed for the achievement of its subgoals possibly via delegation. The actions to be performed on these resources (i.e., access, modify, manage) are derived by the analysis of the goals for which the resource is directly linked. For instance, goal *maintain patient profile* in Figure 2 (and thus goal *update patient profile*) requires 'modify' rights on the patient profile. Goal elicited through the dependence network to increase the conviviality of the system (e.g., *get social interaction* in our scenario) are analyzed in similar way. For instance, the analysis of goal *get social interaction* shows that its achievement requires 'access' rights for social support resources. In this work, we rely on the tool presented in [48], which implements the approach in [51], to automatically infer the list of resources needed to achieve a goal.

Table 3 presents the socio-technical mapping matrix for our scenario. In the table, "+" is used to represent that executing a certain action on a certain resource is

[3] Note that OR decomposition may lead to alternative sets of resources that may be needed to achieve a goal. For the sake of simplicity, we do not address this issue here and refer to [51] for detail.

(Resources,Actions)/ Goals		Stay Healthy	Update Patient Profile	Get Real-Time Data	Provide first aid	Provide Social Support	Get Patient Participation	Get Social Interaction
Patient data	Access	+	+	+	+	NA	NA	NA
	Modify	NA	NA	NA	NA	NA	NA	NA
	Manage	NA	NA	NA	NA	NA	NA	NA
Patient profile	Access	+	+	NA	NA	NA	NA	NA
	Modify	+	+	NA	NA	NA	NA	NA
	Manage	NA	NA	NA	NA	NA	NA	NA
Phone communication system	Access	+	NA	NA	NA	NA	NA	NA
	Modify	NA	NA	NA	NA	NA	NA	NA
	Manage	NA	NA	NA	NA	NA	NA	NA
Social support resources	Access	NA	NA	NA	NA	+	+	+
	Modify	NA	NA	NA	NA	+	+	NA
	Manage	NA	NA	NA	NA	NA	NA	NA

Table 3 Socio-technical mapping matrix

necessary to achieve the goal, and *NA* (i.e., not applicable) to represent that a certain resource (or an action) is not needed for the achievement of the goal.

The analysis of the dependence network may lead to consider potential dependencies between agents to improve the conviviality of the system. However, the impact of such dependencies on the system security should be analyzed. Indeed, dependencies cannot be deployed in the system if they lead to security breaches. To assess the impact of a potential dependency on the access control policy, we identify which authorization rules are needed to achieve the delegated goal using the socio-technical mapping matrix. We refer to those authorizations as *candidate authorization rules*, denoted by *ca*. Given a potential dependency $dep(a, b, s)$ where a is the depender, b the dependee, and s is the agent creating the dependency, the corresponding set of candidate authorization rules $ca(dep(a, b, s))$ is identified as follows:

- For each g such that $\{g \in G | G = dep(a, b, s)\}$, the pairs $(resource, action)$ needed for achievement of g are determined through the socio-technical mapping matrix.
- Each identified pair $(resource, action)$ is augmented with the dependee b. The resulting set forms the set of candidate authorization rules.

Thus, we resort to the socio-technical mapping matrix to identify the candidate authorization rules needed to carry out the duties assigned through a given potential dependency. For example, in the dependence network of Figure 4, goal *get social interaction* involves a depender *Neighbor* and a dependee *Patient*. Through the mapping matrix, the goal is mapped to resource *social support resources* and action *access*. Accordingly, deploying the dependency within the system, while ensuring that the goal can be achieved, requires updating the access control policy with the following authorization rule:

$$\langle Patient, access, social\ support\ resources, permit \rangle$$

Once the candidate authorization rules are identified, the security expert should evaluate them against the current access control policy. In particular, the access control policy is updated only if candidate authorization rules do not conflict with non-negotiable authorization rules (see Section 5). In the next section, we present a strategy for policy update based on the distinction between negotiable and non-negotiable authorization rules.

7.3 Policy Update

To evaluate whether a policy update is eligible, candidate authorization rules should be evaluated against negotiable and non-negotiable authorization rules. To not violate the security constraints, the following strategy is followed to update the policy:

Case 1 (No Conflict): If all candidate authorization rules related to a potential dependency correspond to negotiable authorization rules, then the potential dependency is deployed in the system and the policy is updated by including the candidate authorization rules.

Case 2 (Conflict): If there exists a candidate authorization rule related to a potential dependency which is in conflict with a non-negotiable authorization rule (i.e., the two rules have a different ruling), then the potential dependency is rejected. Conviviality is not increased; however, the system remains secure.

Case 3 (Neutral): If a potential dependency does not require updating the access control policy, then the potential dependency is deployed in the system. Conviviality is increased, and the access control policy remains unchanged. The system remains secure since the deployed dependency has no impact on the security policy.

In the next section, we present the application of policy update to our AAL scenarios.

8 Validation

In this paper, we propose a methodological framework for updating and adapting access control policies based on conviviality recommendations. To validate our approach, we have applied it to the Ambient Assistant Living (AAL) domain and in particular to a number of scenarios which have been emphasized by our partner Luxembourg HotCity. The aim of the validation is to verify whether *our methodology improves the conviviality of the system (thus the user experience), while maintaining an appropriate level of security.* This section first describes two validation scenarios and the prototype implementation of the running example. These descriptions are followed by a presentation of the results in Section 8.3 and by a discussion on the threats to validity in Section 8.4.

8.1 Validation Scenarios

The methodology presented in this paper has been applied to a selection of twelve scenarios. The reader may refer to a technical report [71] for more details concerning the different scenarios besides the authorization rules and the dependence networks related to each scenario. This selection was done by the HotCity experts based on the following two criteria: 1) *likelihood*, i.e., the probability that the scenario occurs and 2) *impact*, i.e., the consequence on human life of the failure of the scenario. Here, we just detail two of them, and present the results of the others in Section 8.3.

The two selected scenarios illustrate how the system is adapted given a new dependency. Each scenario has been modeled with a dependence network. Later, we consider the models to infer the potential goals and dependencies that may be added to increase the number of cycles in the network, i.e., conviviality. For each potential dependency,

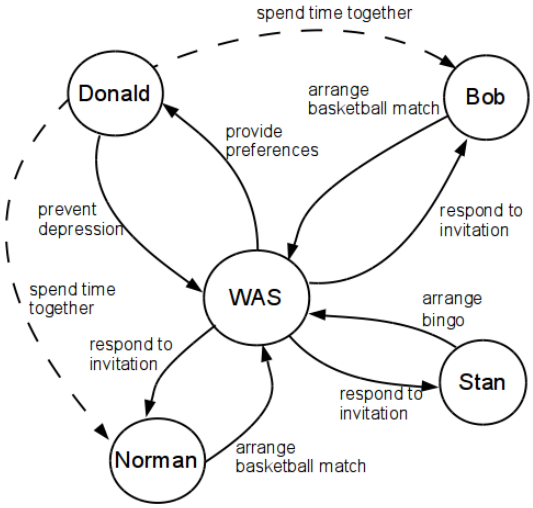

Fig. 6 Dependence Network of Scenario 2: Depression 1

we used socio-technical mapping matrices to infer the changes to the authorization policies governing the scenarios.

8.1.1 Scenario 1: Heart Attack 1

This scenario, which is the running example, has been presented in the dependence network of Figure 3. In this scenario, the neighbor is isolated and does not depend on any another agent in the system. The neighbor may have a potential dependency with the patient, Annette Becker, to get a social interaction like presented in Figure 3. Such a dependency is mapped to authorization rule $\langle Patient, access, social\ support\ resources, permit\rangle$ through the socio-technical mapping matrix in Table 3. Indeed, the fulfillment of goal *get social interaction* requires access to social support resources. Since there are no hard requirements conflicting with this authorization rule, the rule is added to the authorization policy as negotiable rule (see *Case 1* in the previous section). Thus, the conviviality of the system is increased, while the system still complies with the elicited security requirements.

8.1.2 Scenario 2: Depression 1

In this scenario, Donald is a 32 years old salesman who lives alone and has no social activity besides his job. The scenario is illustrated by the dependence network of Figure 6. Donald depends on the Welfare Assistance System (WAS) of the local hospital for goal *prevent depression*. In turn, the WAS depends on Donald to have his preferences concerning social activities he enjoys and his availability. In addition, other patients, namely Norman, Stan and Bob, depend on the WAS for organizing social activities. The WAS uses a scheduling system to organize social activities for patients.

The policy that regulates the scenario includes a number of non-negotiable authorization rules regulating the access to the scheduling system and patients' preferences: R_1: $\langle WAS, manage, scheduling\ system, permit\rangle$

R_2: $\langle Donald, access, scheduling\ system, deny\rangle$
R_3: $\langle Norman, access, scheduling\ system, deny\rangle$
R_4: $\langle Stan, access, scheduling\ system, deny\rangle$
R_5: $\langle Bob, access, scheduling\ system, deny\rangle$
R_6: $\langle Donald, manage, Donald\text{-}preferences, permit\rangle$
R_7: $\langle WAS, access, Donald\text{-}preferences, permit\rangle$
R_8: $\langle Norman, access, Donald\text{-}preferences, deny\rangle$
R_9: $\langle Stan, access, Donald\text{-}preferences, deny\rangle$
R_{10}: $\langle Bob, access, Donald\text{-}preferences, deny\rangle$

To tackle Donald's issues, the WAS aims to synchronize Donald with other patients that have common interests. As Donald expressed interest in playing basketball, a new dependency between him and other patients who also enjoy playing basketball is suggested. These candidate dependencies are denoted with a dashed line in Figure 6.

The deployment of these dependencies have no impact on the access control policy. Indeed, activities are directly organized by the WAS; thus, patients are not required to have access to the scheduling system or to the preferences of other patients. Therefore, the identified candidate dependencies can be deployed in the system, increasing the conviviality of the system while the system remains secure (*Case 3*).

8.2 Implementation

In this section, we present the prototype implementation of our running example scenario. In order to clearly separate the different levels of details, the set of equipment deployed in the house is referred to as Home Care System (HCS). Such an equipment includes actuators, sensors and a controller, Sensors collect information about patient health status. For instance, patients are provided with a watch measuring temperature and pulse rate. The central component of the HCS is the *Home Controller (HC)*. This component is the operation center of the HCS. It is connected to sensors and actuators and makes computations on collected data. Its prototype implementation uses Kevoree as an environment for both design and runtime. Kevoree[4], developed by the Triskell team at University of Rennes 1, makes use of a component-based approach and Models@Runtime to provide a highly dynamic environment.

The HC is composed of several components in charge of the gathering of patient information from sensors, actuators and services surrounding them and their processing. Amongst these components, the *BodySensors* component collects information from the watch; the *VideoRecorder* grabs pictures of the scene in case of need; the *TextMessageModem* sends text messages through a GSM modem. The *Patient Health Record (PHR)* component makes the link with the Electronic Emergency Responder [39] of the local hospital and gets updates of the patient record. The *Emergency Call List* (ECL) stores the name and phone number of the persons to be contacted in case of emergency as defined by DR44 of [39]. The *Workflow Manager (WM)* is responsible of orchestrating the emergency process. In particular, this component takes care of the execution of the sequence of tasks to be executed when an event occurs.

Figure 7 presents the sequence diagram describing the steps executed by the WM. The process is initialized by the BodySensors component when it receives an alert of

[4] http://www.kevoree.org

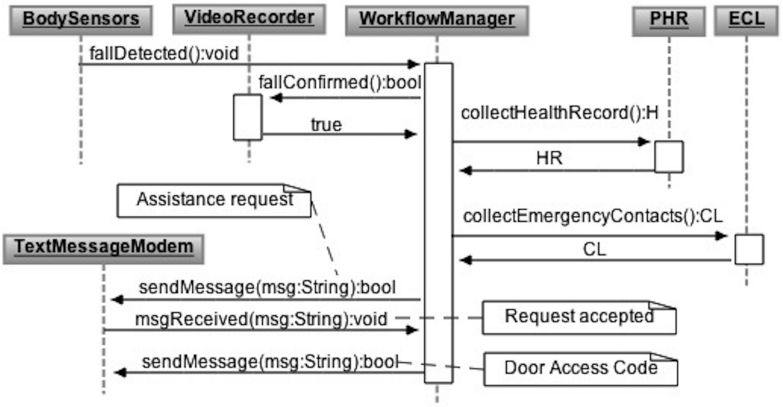

Fig. 7 Scenario Sequence Diagram

a fall along with the data collected by the watch such as temperature and pulse rate. The fall is considered an emergency case; how serious this case is must be evaluated by the HC. To this end, the alert automatically triggers the execution of the WM component. The first step is to confirm the fall using another source of information. The VideoRecorder component collects and processes images captured by video cameras in the house. Once such information has been gathered, the HC can confirm the fall, and resumes the execution of the workflow. The HC needs to collect health information about the patient in order to decide whether the situation is a low, a medium or a high level emergency.

Patients' health information is requested by the PHR to the Electronic Emergency Responder of the local hospital. The data collected are the clinical summary (DR02) and decision support data (DR17) as described in [39]. By compiling all the information collected, the HC makes the decision about the level of emergency.

In case of a medium emergency, the HC has to find someone who can provide assistance to the patient. To this end, the workflow activates the ECL component. The ECL provides the HC with the list of names and phone numbers of people trusted by the patient (family, friends, etc.). The HC contacts each number by sending a text message inquiring their availability in the defined order until someone accepts to assist the patient. The first contact that answers positively receives another text message containing the door access code to enter the flat.

The security of the process is managed by the *Access Control Manager* (ACM). This component is able to reason about the current state of the HC thanks to the Models@Runtime capabilities of Kevoree. Consequently, it works at a higher level of abstraction than the other components and is not visible to them. The ACM ensures that access control policies are properly enforced; it also implements the policy update presented in Section 7. If changes in the HC are detected (e.g., a person is added/removed to the ECL of a patient), the ACM assists operators in the analysis of the conviviality of the updated system. Based on the changes, potential ameliorations can be proposed. Such ameliorations are verified against the access control policy and in particular against non-negotiable authorization rules. If no hard constraints have been violated, the policy is updated and the ACM requests Kevoree to actually adapt the

ID	Scenario title	L	I	$P = L \times I$	Convi-viality before	Convi viality after
1	Heart-attack 1	2.5	3	7.5	2	3
2	Loneliness	3	2	6	4	5
3	Isolation	2	3	6	2	5
4	Finances	2	3	6	3	6
5	Fever	3	2	6	3	5
6	Medication	3	1	3	1	3
7	Weight	3	1	3	1	3
8	Depression 1	3	1	3	4	4
9	Alzheimer	2	1	2	2	5
10	Depression 2	1	2	2	3	5
11	Alcoholism	1	2	2	3	4
12	Heart-attack 2	1	1	1	5	7

Table 4 Prioritization of the scenarios, based on Likelihood and Impact

running system to fit the newly created policy. Accordingly, Kevoree may connect (or disconnect) components to the HC to conform with the updated access control policy.

8.3 Results

This section summarizes the results of our methodology with respect to the 12 scenarios selected by HotCity (Table 4). As in risk based testing approaches [34], likelihood and impact have been used to prioritize scenarios, from low (value 1) to high (value 3). The *priority* P of each scenario is calculated as the product of *likelihood* L and *impact* I, i.e. $P = L \times I$. The results describe the relevancy of our scenarios. Table 4 presents the results of this ranking in a descending order in terms of Priority. ID identifies the scenarios that are considered in the validation. Conviviality is measured by the number of cycles in the dependence network. The result shows that, in the worst case, we have not been able to improve the conviviality adding at least one cycle to the network, thereby one potential coalition among the participating agents. However, we managed to increase the conviviality with our approach in 91.6% of the selected scenarios, with sometimes major improvements (of factor 3). Consequently, it appears that by just combining the social dimension, i.e., conviviality, and access control policy may bring improvements regarding how users perceive the system. This means that conviviality may be effortlessly improved without degrading the security of the system. Indeed, we have only considered changes that do not affect non-negotiable authorization rules. The added-value of our approach is to make explicit decisions that were previously taken in an ad-hoc manner by considering social and security aspects while reconsidering the system design.

8.4 Threats to validity

During this work, we have have identified potential threats to the validity of the proposed approach and its validation. This section lists some of those threats. As a threat to internal validity, we can assume that the methodology we propose requires an expert to manually check for conviviality improvement. We need to go beyond a methodology

definition and find a systematic approach to allow automated self improvement of the system's conviviality while keeping in mind the security policy.

The threats to external validity are related to the level to which our scenarios are representative of real life case studies. The scenarios that we have considered to validate the approach were quite simple in terms of number of actors or goals. In the future, we intend to improve the validation process using more elaborated scenarios with more goals, and more actors, to show the effectiveness of our approach to achieve scalability.

9 Related Work

This work spans four main research areas, namely *assistance in policy specification*, *policy adaptation*, *requirements negotiation* and *conviviality*. In the domain of assistance in policy specification, some contributions have been proposed to fill the gap between requirements analysis and polices specification. Basin et al. [4] present a UML-based modeling language, called SecureUML, for modeling access control policies and integrating these (policies) into a model-driven software development process. Dardenne et al. [28] propose a process for refining requirements and derive security policies from them. In particular, the refinement process allows the derivation of access control policies and obligations expressed in Ponder [27]. Another work in the same direction has been presented by Crook et al. [25] who propose a framework for defining access control policies which considers the assignment of users to the roles within an organization. These proposals, however, focus on the system-to-be, and do not analyze the organizational environment in which the system will eventually operate. In particular, they do not consider the social relations between stakeholders which are the basis for specifying conviviality-driven access control policies. Massacci et al. [46,47] present a quantitative approach to determine the access control policy for an inter-organizational business process, which is minimal with respect to the sensitivity of data and the level of trust between actors. This approach allows users to express their preferences in the form of privacy penalties associated to personal data and to the partner of the business process. Then, it determines the alternative with the smallest privacy penalty and thus guarantees maximal privacy protection. In contrast, our work mainly focuses on the trade-off between conviviality and security, where the number and size of coalitions is the main criterion to evaluate the conviviality of the system and access control policies comply with the need-to-know principle by construction.

Several research efforts have addressed policy adaptation in dynamic environments. Rinderle-Ma and Reichert [57] propose a formal framework for modeling changes in the organizational models and in the corresponding access control policy. Bertino et al. [6] present a model in which users can dynamically change their access requests to obtain authorizations. Requests include service parameters and subject partial identities to establish trust using trust negotiation. Ray [55] proposes concurrency control algorithms to allow real-time and concurrent policy updates in a database system. Ryutov et al. [59] propose a framework to support policy adaptation based on suspicion level and system threat level. Lymberopoulus et al. [45] present a framework for policy specification and management in network services. In their framework, policies adaptation is monitored by event triggers that allow the change of a policy on the basis of the changes in the managed environment. Morin et al. [53] introduce a model-based approaches to update the architecture model according to security policy. Here, a domain specific modeling language is used to establish a mapping between the access control policy

and the architecture model. Research efforts on policy adaptation can be categorized based on the nature of adaptation and changes; to the best of our knowledge, our work is the first work that considers dependence relations between agents to update access control policies.

In the domain of requirements negotiation and trade-offs, Boehm and Egyed [7] present a process that permits to capture, analyze and negotiate requirements in order to satisfy the higher number of stakeholders. Kazman et al. [42] propose a spiral model of design to help identify and understand the trade-offs inherent in the architectures of systems that contain competing quality attributes. Their methodology helps identify dependencies among different attributes (trade-off points) and permits reasoning about them. Several research initiatives [2,11,26,32,33,38,61,72] have tackled trade-offs issues between security and social dimensions of socio-technical systems and in particular the trade-off between security and usability. For instance, Yee [72] has proposed to align security and usability by considering users' workflow and deriving authorizations from users' tasks. Braz et al. [11] explore trade-offs between security and usability through usability inspection methods based on automata machines theory. The method provides criteria for studying usability factors which are refined into usability metrics for a secure system. Flechais et al. [32,33] propose a methodology supported by well defined semantics that considers security, risk analysis and the context of use at design phase of a software development to achieve a trade-off between security and usability. The approach is based on an asset model that is extended with contextual information about the system usability. Our work is orthogonal to those proposals in the fact that we study the trade-off between security and another social dimension of socio-technical systems, namely conviviality.

Similarly to our work, the work by Liu et al. [44] and subsequent work by Elahi et al. [29,30] extend i* to capture security and privacy requirements and enable trade-off analysis between security and other non-functional requirements. Intuitively, security is treated as a non-functional requirement: softgoals, as "Security" or "Privacy", are used to model the corresponding notions. In addition, an attacker model is constructed within the requirements model, and dependency analysis is used to determine the level of security guaranteed by the system by analyzing the satisfaction of the corresponding softgoals. Although this approach makes it possible to assess the risk of security incidents and evaluate the impact of countermeasures on the systems, it is not suitable to derive an access control policy from the requirements model. Bryl et al. [13] propose a requirements analysis approach for socio-technical systems which employs planning techniques for exploring the space of requirements alternatives and a number of social criteria for their evaluation. This approach has also been applied to SI* to select the optimal security design among a set of alternatives [14]. The plan obtained using the approach in [13,14] are optimal with respect to the length of the plan, where optimality is defined in terms of length minimization. This approach, however, is not applicable to conviviality since the plan with minimal length is usually not the one that maximizes conviviality. Bryl et al. [13] also propose metrics to study the criticality of an actor in a plan. Our approach would benefit from the application of such metrics to dependence networks as they provide insights on the resilience of dependence networks.

Conviviality has been introduced as a social concept in multi-agent systems that reflects relations between individuals to emphasize some human aspects like equality and community life [22]. In previous studies (e.g., [16]) conviviality is measured in terms of interdependencies between agents. The basis idea is that more opportunities to work with other people increases the conviviality, whereas larger coalitions may de-

crease the efficiency and stability of these coalitions. Our work considers conviviality from a different perspective: conviviality can be increased as long as it does not impact the system security. Conviviality has been captured through three models using dependence networks [19]: the first model captures temporal properties to reason about conviviality evolution over time; the second model captures stakeholders viewpoints; and the third model captures transformations of social dependencies by hiding power relations and social structures to facilitate social interactions. In our work, we do not consider the temporal dimension that may regulate agents' dependencies; this aspect will be investigated in future work.

10 Conclusions and Future Work

Changes in socio-technical requirements, design, and environment may require to adapt and update the access control policy regulating the system. This paper presents the DN-AC alignment methodology for analyzing access control policies with respect to the concept of conviviality. We have used a goal-oriented methodology to capture and analyze the social interactions between stakeholders. Then, security interactions are used to define the access control policy, whereas dependencies are used to analyze, through dependence networks, the conviviality of the system. To reconcile the security and conviviality visions, we proposed a socio-technical mapping matrix that connects concepts of access control and concepts related to dependence networks to analyze the impact of conviviality on existing authorization rules. We also defined how to adapt authorization rules based on the impact of conviviality on the system security. To validate the proposed methodology, we have built a proof-of-concept prototype from the AAL use case of Luxembourg HotCity. The main lesson learned from the scenario is that the outcome of our methodology leads to create more coalitions between agents and thus to increase the conviviality while maintaining the security level of the system.

Further works involve refining the process of automatic derivation of dependence networks and AC policy from requirements. This will enable to systematically analyze complex scenarios involving a large number of agents and dependencies. Moreover, in this paper we consider static models, whereas investigating the evolution of models would provide finer-grained analysis over the conviviality improvement and AC policy updating process. Finally, we need to implement a prototype of the mapping approach to automate the process of policy update while considering the potential dependencies in the system with the objective to improve conviviality. This automation is needed to provide security experts with an automatic reasoning tool for policy update.

References

1. eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, OASIS (2012). URL `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf`
2. Ackerman, M.: Usability and security. In: Proceedings of the Network and Distributed System Security Symposium. The Internet Society (1999)
3. Asnar, Y., Li, T., Massacci, F., Paci, F.: Computer Aided Threat Identification. In: Proceedings of 13th IEEE Conference on Commerce and Enterprise Computing, pp. 145–152. IEEE (2011)
4. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From uml models to access control infrastructures. ACM Trans. Softw. Eng. Methodol. **15**(1), 39–91 (2006)

5. Baxter, G., Sommerville, I.: Socio-technical systems: From design methods to systems engineering. Interacting with Computers **23**(1), 4–17 (2011)
6. Bertino, E., Squicciarini, A.C., Martino, L., Paci, F.: An adaptive access control model for web services. Int. J. Web Service Res. **3**(3), 27–60 (2006)
7. Boehm, B., Egyed, A.: Software requirements negotiation: some lessons learned. In: Proceedings of the 20th International Conference on Software Engineering, pp. 503–506. IEEE (1998)
8. Boella, G., Sauro, L., van der Torre, L.W.N.: Social viewpoints on multiagent systems. In: AAMAS, pp. 1358–1359. IEEE Computer Society (2004)
9. Boella, G., van der Torre, L., Villata, S.: Four ways to change coalitions: Agents, dependencies, norms and internal dynamics. In: Proceedings of the 2nd Multi-Agent Logics, Languages, and Organisations Federated Workshops, *CEUR Workshop Proceedings*, vol. 494. CEUR-WS.org (2009)
10. Bonzon, E., Lagasquie-Schiex, M.C., Lang, J.: Dependencies between players in boolean games. Int. J. Approx. Reasoning **50**(6), 899–914 (2009)
11. Braz, C., Seffah, A., M'Raihi, D.: Designing a trade-off between usability and security: A metrics based-model. In: Human-Computer Interaction–INTERACT 2007, LNCS 4663, pp. 114–126. Springer (2007)
12. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems **8**(3), 203–236 (2004)
13. Bryl, V., Giorgini, P., Mylopoulos, J.: Designing socio-technical systems: from stakeholder goals to social networks. Requir. Eng. **14**(1), 47–70 (2009)
14. Bryl, V., Massacci, F., Mylopoulos, J., Zannone, N.: Designing security requirements models through planning. In: Proceedings of 18th International Conference on Advanced Information Systems Engineering, LNCS 4001, pp. 33–47. Springer (2006)
15. Caire, P.: New tools for conviviality: Masks, norms, ontology, requirements and measures. Ph.D. thesis, Luxembourg University, Luxembourg (2010)
16. Caire, P., Alcade, B., van der Torre, L., Sombattheera, C.: Conviviality measures. In: Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 895–902. International Foundation for Autonomous Agents and Multiagent Systems (2011)
17. Caire, P., Bikakis, A., Efthymiou, V.: Conviviality by Design. In: Proceedings of Symposium on Social Computing - Social Cognition - Social Networks and Multiagent Systems (2012)
18. Caire, P., van der Torre, L.: Convivial ambient technologies: Requirements, ontology and design. The Computer Journal **53**(8), 1229–1256 (2009)
19. Caire, P., van der Torre, L.: A conviviality measure for early requirement phase of multiagent system design. In: Normative Multiagent Systems, no. 09121 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009)
20. Caire, P., van der Torre, L.: Temporal dependence networks for the design of convivial multiagent systems. In: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1317–1318. International Foundation for Autonomous Agents and Multiagent Systems (2009)
21. Caire, P., Villata, S., Boella, G., van der Torre, L.: Conviviality masks in multiagent systems. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1265–1268. International Foundation for Autonomous Agents and Multiagent Systems (2008)
22. Caire, P., Villata, S., Boella, G., van der Torre, L.: Conviviality masks in multiagent systems. In: Proceedings of the 7th International Joint Conference on Autonomous Agents And Multiagent Systems, pp. 1265–1268. International Foundation for Autonomous Agents and Multiagent Systems (2008)
23. Castelfranchi, C.: The micro-macro constitution of power. Protosociology **18**, 208–269 (2003)
24. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press (2001)
25. Crook, R., Ince, D., Nuseibeh, B.: Modelling access policies using roles in requirements engineering. Information and Software Technology **45**(14), 979–991 (2003)
26. Damen, S., Zannone, N.: Privacy Implications of Privacy Settings and Tagging in Facebook. In: Proceedings of the 10th VLDB Workshop on Secure Data Management. Springer (2013)

27. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, LNCS 1995, pp. 18–38. Springer (2001)
28. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. In: Proceedings of the 6th International Workshop on Software Specification and Design, pp. 3–50. Elsevier Science Publishers B. V., Amsterdam, The Netherlands (1993)
29. Elahi, G., Yu, E.S.K.: Modeling and analysis of security trade-offs - a goal oriented approach. Data Knowl. Eng. **68**(7), 579–598 (2009)
30. Elahi, G., Yu, E.S.K., Zannone, N.: A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. Requir. Eng. **15**(1), 41–62 (2010)
31. Emerson, R.: Power-dependence relations. American Sociological Review **27**, 31–41 (1962)
32. Flechais, I., Mascolo, C., Sasse, M.A.: Integrating security and usability into the requirements and design process. Int. J. Electron. Secur. Digit. Forensic **1**(1), 12–26 (2007)
33. Flechais, I., Sasse, M.A., Hailes, S.M.V.: Bringing security home: a process for developing secure and usable systems. In: Proceedings of the 2003 Workshop on New Security Paradigms, pp. 49–57. ACM (2003)
34. Frankl, P.G., Weyuker, E.J.: Testing software to detect and reduce risk. Journal of Systems and Software **53**(3), 275–286 (2000)
35. Fuxman, A., Liu, L., Mylopoulos, J., Roveri, M., Traverso, P.: Specifying and analyzing early requirements in Tropos. Requir. Eng. **9**(2), 132–150 (2004)
36. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling Security Requirements Through Ownership, Permission and Delegation. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, pp. 167–176. IEEE Computer Society (2005)
37. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Requirements engineering for trust management: model, methodology, and reasoning. Int. J. Inf. Sec. **5**(4), 257–274 (2006)
38. Gutmann, P., Grigg, I.: Security usability. Security & Privacy, IEEE **3**(4), 56–58 (2005)
39. Healthcare Information Technology Standards Panel (HITSP): Emergency Responder Electronic Health Record Interoperability Specification (IS04), Version 2.0 (2008)
40. Illich, I.: Tools for Conviviality. Marion Boyars Publishers, London (1974)
41. Jureta, I.J., Mylopoulos, J., Faulkner, S., Schobbens, P.Y.: Core ontology for requirements engineering. Tech. rep., Information Management Research Unit, University of Namur (2007)
42. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The architecture tradeoff analysis method. In: Proceedings of the 4th IEEE International Conference on Engineering of Complex Computer Systems, pp. 68–78. IEEE Computer Society (1998)
43. Koller, D., Milch, B.: Multi-agent influence diagrams for representing and solving games. Games and Economic Behavior **45**(1), 181–221 (2003)
44. Liu, L., Yu, E.S.K., Mylopoulos, J.: Security and Privacy Requirements Analysis within a Social Setting. In: Proceedings of 11th IEEE International Requirements Engineering Conference, pp. 151–161. IEEE Computer Society (2003)
45. Lymberopoulos, L., Lupu, E., Sloman, M.: An adaptive policy-based framework for network services management. J. Netw. Syst. Manage. **11**, 277–303 (2003)
46. Massacci, F., Mylopoulos, J., Zannone, N.: Minimal Disclosure in Hierarchical Hippocratic Databases with Delegation. In: Proceedings of 10th European Symposium on Research in Computer Security, LNCS 3679, pp. 438–454. Springer (2005)
47. Massacci, F., Mylopoulos, J., Zannone, N.: Hierarchical hippocratic databases with minimal disclosure for virtual organizations. VLDB J. **15**(4), 370–387 (2006)
48. Massacci, F., Mylopoulos, J., Zannone, N.: Computer-aided Support for Secure Tropos. Autom. Softw. Eng. **14**(3), 341–364 (2007)
49. Massacci, F., Mylopoulos, J., Zannone, N.: An ontology for secure socio-technical systems. In: Handbook of Ontologies for Business Interaction, pp. 188–207. IDEA Group (2007)
50. Massacci, F., Mylopoulos, J., Zannone, N.: Security requirements engineering: The si* modeling language and the secure tropos methodology. In: Z.W. Ras, L.S. Tsay (eds.) Advances in Intelligent Information Systems, *Studies in Computational Intelligence*, vol. 265, pp. 147–174. Springer (2010). URL `http://dblp.uni-trier.de/db/series/sci/sci265.html#MassacciMZ10`
51. Massacci, F., Zannone, N.: A model-driven approach for the specification and analysis of access control policies. In: Proceedings of OTM Confederated International Conferences, LNCS 5332, pp. 1087–1103. Springer (2008)

52. Montali, M., Torroni, P., Zannone, N., Mello, P., Bryl, V.: Engineering and verifying agent-oriented requirements augmented by business constraints with *B*-Tropos. Autonomous Agents and Multi-Agent Systems **23**(2), 193–223 (2011)
53. Morin, B., Mouelhi, T., Fleurey, F., Le Traon, Y., Barais, O., Jézéquel, J.M.: Security-driven model-based dynamic adaptation. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 205–214. ACM (2010)
54. NeOn: NeOn Toolkit. http://neon-toolkit.org/
55. Ray, I.: Applying semantic knowledge to real-time update of access control policies. IEEE Trans. on Knowl. and Data Eng. **17**(6), 844–858 (2005)
56. Ray, I., France, R., Li, N., Georg, G.: An aspect-based approach to modeling access control concerns. Information and Software Technology **46**, 575–587 (2004)
57. Rinderle-Ma, S., Reichert, M.: A formal framework for adaptive access control models. Journal on Data Semantics IX pp. 82–112 (2007)
58. Ruscio, D.D., Muccini, H., Pierantonio, A., Pelliccione, P.: Towards weaving software architecture models. In: Proceedings of International Workshop on Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, pp. 103–112. IEEE Computer Society (2006)
59. Ryutov, T., Zhou, L., Neuman, C., et al.: Adaptive trust negotiation and access control. In: Proceedings of the 10th ACM Symposium on Access Control Models And Technologies, pp. 139–146. ACM (2005)
60. Saltzer, J., Schroeder, M.: The protection of information in computer systems. Proceedings of the IEEE **63**(9), 1278–1308 (1975)
61. Sasse, M.A., Flechais, I.: Usable security: Why do we need it? how do we get it? In: Security and Usability: Designing secure systems that people can use, pp. 13–30. O'Reilly (2005)
62. Sauro, L.: Qualitative criteria of admissibility for enforced agreements. CMOT **12**(2-3), 147–168 (2006)
63. Sauro, L., Villata, S.: Dependency in cooperative boolean games. J Logic Comp. **23**, 425–444 (2013)
64. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. John Wiley & Sons (2004)
65. Sharman, R., Kishore, R., Ramesh, R.: Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems (Integrated Series in Information Systems). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
66. Sichman, J.S.: Depint: Dependence-based coalition formation in an open multi-agent scenario. Journal of Artificial Societies and Social Simulation **1**(2), 1998 (1998)
67. Sichman, J.S., Conte, R.: Multi-agent dependence by dependence graphs. In: Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems, pp. 483–490. ACM (2002)
68. Sichman, J.S., Demazeau, Y.: On social reasoning in multi-agent systems. Revista Iberoamericana de Inteligencia Artificial **13**, 68–84 (2001)
69. Sinclair, S., Smith, S.W.: What's wrong with access control in the real world? IEEE Security and Privacy **8**, 74–77 (2010)
70. Torasso, D.P., Inf, A.: Formalizing admissibility criteria in coalition formation among goal directed agents. Ph.D. thesis (2006). URL `http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.100.4550`
71. Vasileios Efthymiou, P.C.: Diagram Analysis Report: Use Cases for Conviviality and Privacy in Ambient Intelligent Systems. University of Luxembourg, SnT, Luxembourg (2012)
72. Yee, K.P.: Aligning security and usability. Security & Privacy, IEEE **2**(5), 48–55 (2004)
73. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto, Canada (1995)