A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments

Alessandra Toninelli[1], Rebecca Montanari[1], Lalana Kagal[2], and Ora Lassila[3]

[1] Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - Italy
{atoninelli, rmontanari}@deis.unibo.it
[2] MIT CSAIL
32 Vassar Street, Cambridge, MA 02139, USA
lkagal@csail.mit.edu
[3] Nokia Research Center Cambridge
3 Cambridge Center, Cambridge, MA 02142, USA
ora.lassila@nokia.com

Abstract. Wireless connectivity and widespread diffusion of portable devices offer novel opportunities for users to share resources anywhere and anytime, and to form ad-hoc coalitions. Resource access control is crucial to leverage these ad-hoc collaborations. In pervasive scenarios, however, collaborating entities cannot be predetermined and resource availability frequently varies, even unpredictably, due to user/device mobility, thus complicating resource access control. Access control policies cannot be defined based on entity's identities/roles, as in traditional access control solutions, or be specified a priori to face any operative run time condition, but require continuous adjustments to adapt to the current situation. To address these issues, this paper advocates the adoption of novel access control policy models that follow two main design guidelines: *context-awareness* to control resource access on the basis of context visibility and to enable dynamic adaptation of policies depending on context changes, and *semantic technologies* for context/policy specification to allow high-level description and reasoning about context and policies. The paper also describes the design of a semantic context-aware policy model that adopts ontologies and rules to express context and context-aware access control policies and supports policy adaptation.

1 Introduction

Telecommunication systems and the Internet are converging towards an integrated pervasive scenario that permits users to access services anytime and anywhere even when they are on the move. Recent technological advances in both computational capabilities and connectivity of portable devices are also enabling mobile users in physical proximity of each other to form ad-hoc networks for spontaneous coalitions and to engage in opportunistic and temporary resource sharing without relying on the availability of a fixed network infrastructure.

I. Cruz et al. (Eds.): ISWC 2006, LNCS 4273, pp. 473 – 486, 2006.

However, these ad-hoc collaborations impose several challenges to the secure retrieval of and operation on distributed resources, undermining several assumptions of traditional access control solutions. These solutions usually assign permissions to principals depending on their identity/role. In the new pervasive scenario, however, users typically share services with unknown entities and, more importantly, with entities whose identity may not be sufficiently trustworthy. In addition, since spontaneous collaborations among users are typically established in an impromptu and opportunistic fashion, it may not be possible to rely on formal collaboration agreements to decide who can access which resources and how, thus excluding the possibility to exploit access control policies defined on a contractual basis as in medium or long-term inter-organizational coalitions. Access control in spontaneous coalitions is further complicated by the high dynamicity in resource availability. Each collaborating entity may alternatively play the role of either a service client or provider or both, depending on dynamic conditions and the current status of interaction. When playing the service provider role, an entity may introduce new services into the environment, thus changing the set of available resources. Variations in resource availability occur also because of the transience of ad-hoc coalitions where entities -resource providers- leave and/or enter a coalition, unpredictably, at any time.

Appropriate access control models are needed to enable resource sharing and access in spontaneous coalition scenarios. It is crucial that the definition and enforcement of access control policies take into account the heterogeneity and dynamicity of the environment in terms of available services, computing devices, and user characteristics. To address these issues, this paper advocates a paradigm shift from subject-centric access control models to context-centric ones. Hereinafter, at a high level, the term "context" is defined as any information that is useful for characterizing the state or the activity of an entity or the world in which this entity operates [1]. Differently from subject-centric solutions where context is an optional element of policy definition that is simply used to restrict the applicability scope of the permissions assigned to the subject, in context-centric solutions, context is the first-class principle that explicitly guides both policy specification and enforcement process and it is not possible to define a policy without the explicit specification of the context that makes policy valid. We also claim that context-centric access control solutions need to adopt ontological technologies as key building blocks for supporting expressive policy modeling and reasoning. Semantically-rich policy representations permit description of policies at different levels of abstraction and support reasoning about both the structure and properties of the elements that constitute a pervasive system, i.e., the context and the management policies, thus enabling policy analysis, conflict detection, and harmonization.

This paper describes an implementation of these ideas in a policy model that exploits context-awareness and ontological technologies for the specification and the evaluation of access control policies. In our access control framework the role of context exploitation for controlling access control is twofold. Drawing inspiration from the RBAC model that exploits the concept of role as a mechanism for grouping subjects based on their properties [2], we state that, the same as with role, the concept of context can provide a level of indirection between entities requesting resource access and their permitted set of actions on requested resources. Instead of assigning

permissions directly to the subjects and defining the contexts in which these permissions should be considered valid and applicable, a system administrator defines for each resource the contextual conditions that enable one to operate on it. When an entity operates in a specific context, she automatically acquires the ability to perform the set of actions permitted in the current context.

In addition, we consider context crucial for enabling policy adaptation. In pervasive environments the conditions that characterize interactions between users and resources may be largely unpredictable. Consequently, policies cannot all be specified *a priori* to face any operative run-time situations, but may require dynamic adjustments to be able to control access to resources. We use the term "policy adaptation" to describe the ability of the policy-based management system to adjust policy specifications and evaluation mechanisms in order to enable their enforcement in different, possibly unforeseen situations. In this scope, it is crucial to be able to represent the various operative conditions under which policies should be applied, i.e., the context, and to define the expected behaviour of the policy framework on the basis of such context variations.

Another fundamental design guideline of our access control model is the adoption of an ontological approach using Description Logic (DL) to context/policy specification to enable context/policy classification, comparison, and static conflict detection. We also adopt a rule-based approach taking the perspective of Logic Programming (LP) to encode rules that allows policy makers to specify policies based on context variables whose value is unknown at policy definition time, thus enabling the efficient enforcement of policies defined over dynamically determined context values. Let us note that our work does not aim at providing a unifying logical framework for DL and LP, which have well-known crucial logical mismatches, but rather at combining the logical results obtained by means of their respective reasoning features.

The paper is organized as follows. Section 2 outlines some crucial requirements for the definition of access control policies in dynamic scenarios like inter-organizational spontaneous coalitions. Section 3 presents our proposed semantic context-aware policy model, while Section 4 compares it with related state-of-the art access control solutions. Final remarks and future activities follow in Section 5.

2 Policy Requirements for Spontaneous Coalition Scenarios

To point out some unique challenges in dynamic mobile environments, we start by considering the spontaneous coalition scenario of a meeting occurring during a conference among members of different universities working on a common project. In the remainder of the paper, we use this meeting scenario as a running example to illustrate the main access control challenges and our solution guidelines. In this meeting scenario, each participant may wish to grant access to her resources to other participants, in order to enable cooperation and knowledge sharing. Access to personal resources must be regulated in order to protect them from malicious access or misuse. However, the specification of adequate access control policies in the depicted scenario presents us with several challenges. For example, the complete list

of participants may not be known in advance or it may be modified just before the meeting starts or even during a meeting, thus making it infeasible to define access control policies based on the requestor's identity.

Even the role-based approach seems cumbersome in controlling access to cross-organizational resources, since role definitions and hierarchies might vary across parties, thus making their interpretation difficult outside the specific boundaries of each organization. A possible solution might be the creation of a common ad-hoc role for all meeting participants, to which each participant delegates her roles, so that others are able to access her resources [3]. However, since roles required to access resources have to be separately assigned by each participant to this ad-hoc role, inconsistencies may arise between the access rights of the different members, e.g., in the case of a member being allowed to access another member's resources, but not vice versa. Moreover, the activation/deactivation of such temporary roles represents a critical security issue.

In order to properly control access to resources, we claim the need for a more general and comprehensive approach that exploits not only identity and role information but also other contextual information, such as location, time, ongoing activities, etc. In particular, we believe that it may be advantageous for each participant to define the access control policies for his managed resources simply according to the current conditions of the requestor, the resource, and of the surrounding environment, i.e., the current resource context. For instance, in an informal meeting, access should be granted to those who are currently located in the same room where the resource owner is located, if they actually participate in the activity/project relating to the meeting, as long as current time corresponds to the time scheduled for the meeting. Access control policies should be associated with the combination of one or more context conditions and users should be instantaneously granted/denied access to resources on the basis of those specific context conditions.

The integration of access control with contextual information has two main characteristics. First, it is an example of an active access control model [4]. Active security models are aware of the context associated with an ongoing activity in providing access control and thus distinguish the passive concept of permission assignment from the active concept of context-based permission activation. Second, the exploitation of context as a mechanism for grouping policies and for evaluating applicable ones simplifies access control management by increasing policy specification reuse and by making policy update and revocation easier. In fact, in subject-based access control solutions, the tight coupling of the identities/roles of principals with their permissions and with the operating conditions in the system to grant permitted actions requires security administrators to foresee all contexts in which each principal is likely to operate. In pervasive environments where principals are typically unknown and where contextual conditions frequently change, this traditional approach may lead to a combinatorial explosion of the number of policies to be written, force a long development time, and even introduce potential bugs. The traditional approach, when applied to pervasive scenarios, also lacks flexibility. New access control policies need to be designed and implemented from scratch for any principal when new context situations occur. In a context-centric access control approach, instead of managing

principals and their permissions individually, administrators define the set of permitted actions for each context. When a principal operates in a specific context, the evaluation process of his permissions in that context is triggered.

Another difficulty in dynamic collaboration scenarios is that it is impossible to define in advance all necessary policies for all possible situations. These environments should permit new policies to be dynamically and easily specified on demand as new situations occur as well as allow existing policies to be adapted to meet changing conditions. For example, let us consider the case of a meeting that continues beyond its originally scheduled end time. It is essential to ensure that meeting participants can continue to access each other's resources as long as the meeting is actually taking place. It is therefore necessary to adapt previous policies to reflect the new conditions of the meeting. In the absence of policy adaptation support, access to the policy owner's resources would be denied after the scheduled time, since the conditions that limit the applicability of the policy, specifically the condition concerning time, would be evaluated to be false. In a traditional approach, the policy owner would have to specify another policy to grant access to her resources after the scheduled end time of the meeting. However, this solution presents several disadvantages. First, the resource owner might not be the policy administrator of her resources, and might be unable to specify the policy when needed. In addition, the specification of ad-hoc policies is not a correct approach to policy definition because it does not favor clarity or traceability, thus complicating policy management. Finally, in such a case, efficiency and security might collide. If the policy owner specifies an access control policy that grants access to her resources for a short time interval, e.g., ten minutes, she might possibly be forced to specify the same policy several times because the eventual end time of the meeting is not known in advance. Conversely, a policy granting access for a longer period might allow undesired access to the user's resources after the meeting.

This simple example demonstrates the need for a new approach to policy specification that not only defines policies based on context information, but also allows the seamless adaptation of policies depending on current context. In this example, we need to "instruct" the system such that, if certain context conditions hold, the context activating the policy is still considered active. Essential for policy adaptation is appropriate modeling of contextual information that enables the policy framework to sense and reason about the current situation. This ensures adequate access control even in changing and possibly unforeseen conditions.

Another important principle is the adoption of semantically-rich representations for policy definition. A semantics-based approach allows description of contexts and associated policies at a high level of abstraction, in a form that enables their classification and comparison. This feature is essential, for instance, in order to detect conflicts between policies before they are actually enforced. In addition, semantic techniques can provide the reasoning features needed to deduce new information from existing knowledge. This ability may be exploited by the policy framework when faced with unexpected situations to react in a contextually appropriate way.

3 A Semantic Context-Aware Access Control Policy Model

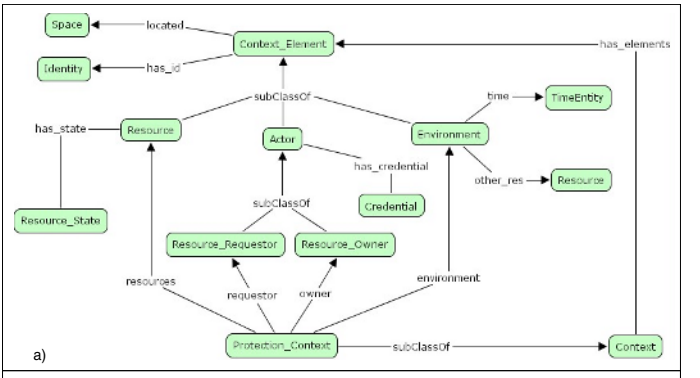
Our access control model is centered around the concept of context that we consider to be any characterizing information about the controlled resources and about the world surrounding them. We adopt a resource-centric approach to context modeling: contexts are associated with the resources to be controlled and represent all and only those conditions that enable access to the resources. Contexts act as intermediaries between the entities requesting access to resources and the set of operations that can be performed on these resources. Access control policies define for each context how to operate on the associated resource(s). In particular, access control policies can be viewed as one-to-one associations between contexts and allowed actions. Drawing inspiration from Java protection domains [5], we call these contexts hereinafter as *protection contexts*: they provide users with a controlled visibility of the considered resource in terms of performable access actions on it (action view). Protection contexts are determined by the defined policies. Entities can perform only those actions that are associated with the protection contexts currently in effect (*active context*), i.e., the contexts whose defining conditions match the operating conditions of the requesting entity, requested resource, and environment as measured by specific sensors. All entities sharing the same active protection context share the same abilities to operate on the context-related resource.

3.1 Context Model

A protection context consists of all the characterizing information that is considered relevant for access control, logically organized in parts that describe the state of the resource associated with the protection context, such as availability or load (the *resource* part), the entities operating on the resource (the policy/resource owner and the requestor), such as their roles, identities or security credentials (the *actor* part), and the surrounding environment conditions, such as time, or other available resources (the *environment* part).

A protection context is a set of attributes and predetermined values, labelled in some meaningful way and associated with desirable semantics [6]. Instead of a single value, an attribute could also define constraints for a range of allowed values. Let us note that an attribute value can be assigned to a fixed constant or can be a variable over a value domain. The current state of the surrounding world is also represented in terms of attribute/value pairs where the attribute values represent the output of sensors (with the term “sensor” used loosely). For a protection context to be “in effect”, the attribute values that define the current state of the world have to match the definition of the context (as given above).

We adopt description logics (DL) and associated inferencing to model and process protection context data. In particular, we use Web Ontology Language (OWL) -based ontologies as shown in Figure 1a. A protection context is defined as a subclass of a generic context and consists of the resource, the actor and the environment context elements. Each context element is characterized by an identity property and a location property defining the physical or logical position of an entity. Single context elements are characterized by specific additional properties.



b) **Meeting Context Specification**

```
Meeting_Context ≡ Protection_Context ⊓ ∃owner.Meeting_Actor ⊓
                  ∃requestor.Co-located_Meeting_Actor ⊓ ∃environment.Meeting_Env ⊓
                  ∃resource.Current_Project_Resource

Meeting_Actor ≡ Actor ⊓ ∃is_currently_working_on.Current_Project ⊓
                ∃located.Meeting_Space ⊓ ∃is_involved_in.Current_Project

Meeting_Env ≡ Environment ⊓ ∃time.In_Current_Meeting_Time

Current_Project_Resource ≡ Project_Resource ⊓
                           ∃is_resource_of_project.Current_Project

Co-located_Meeting_Actor ≡ ... (see Table 1)
```

Fig. 1. Context ontology model and an OWL context specification example

Figure 1b shows an OWL-based protection context representation example related to the meeting scenario depicted in Section 2. This example assumes that each actor taking part to the meeting owns a set of resources that relates to the project/activity the meeting is about and shares these resources with the other participants. In particular, the protection context shown in Figure 1b grants access to these resources under certain conditions: the resources must be specifically pertaining the project discussed at the current meeting; the resource owner must be involved in the meeting's project as "project partner", must be currently work on the project-related set of resources, and must be located in the place where the meeting is planned to take place to guarantee that he is attending the meeting. The entities requesting access to resources must be involved in the project as "project partners", co-located with the resource owner, and currently working on project-specific resources on their devices. In addition, resources can be accessed when the time in the environment corresponds to the time scheduled for the meeting. Let us note that the core context ontology has been extended to model the specific meeting-related concepts. For example, a resource is associated with the project it relates to, an actor has attributes describing the project she is involved in or she is currently working on, and the environment time can be expressed in terms of scheduled events in an actor's calendar. The meeting ontology also explicitly defines the concept of "current event", which is an event or activity occurring at the moment of context and policy evaluation. In addition, we make use of a location ontology that is provided within the basic context model[1].

[1] All our ontologies are available at http://lia.deis.unibo.it/research/SemanticPolicies.

Let us note that the use of DL in context modeling and reasoning has well-known benefits. For instance, considering protection contexts as classes and a set of sensor inputs (i.e., the current state of the world) as individuals, DL-based reasoning allows one to determine which protection contexts are in effect by verifying which protection context classes the current state is an instance of, and to figure out how defined protection contexts relate to each other (nesting, etc.) [6].

However, DL-based reasoning may not always be sufficient. Our context-aware access control model needs more expressive context reasoning in order to be effective. On the one hand, we need to correlate contexts using not only class definitions (as in pure DL-based reasoning) but also property path relationships between anonymous individuals. For instance, in a meeting context we need to state that if the resource owner is located in a certain place and the resource requestor is located in the same place, the two are co-located. On the other hand, we need to bind the context attribute values to specific instances depending on application-specific context attribute/value relationships. For instance, to enforce the meeting-related policies, we must be able to determine, at each moment, what the actual current project is, so that the corresponding resources belonging to each actor are identified and protected. To overcome some DL-based reasoning restrictions we combine it with LP-based reasoning. In particular, we define two types of rules: context *aggregation rules* to support reasoning using property path relationships and context *instantiation rules* to provide OWL assertions for attribute values. For instance, the condition of co-location between two collaborating entities at a conference is expressed with an aggregation rule, whereas the condition of current project with an instantiation rule. Both types of rules are expressed according to the following pattern:

`if` context attributes $C_1 \ldots C_n$ `then` context attribute C_m

that corresponds to a Horn clause, where predicates in the head and in the body are represented by classes and properties defined in the context and application-specific ontologies.

3.2 Context-Aware Access Control Policy Model

Our policy model consists of three distinct phases (see Figure 2a): policy specification, policy refinement, and policy evaluation. In the *policy specification* phase resource administrators specify OWL-based policies representing ontological associations between actions and protection contexts ontology definitions. Figure 2b shows an example of a policy that controls access to the meeting resources. The protection contexts may have attribute values assigned to constants or may be variables. In the latter case, attributes are assigned proper values by combining DL-based and LP-based reasoning over the context ontology and the context aggregation and activation rules. In particular, the output of LP rules is fed into the DL knowledge base to determine the value of each attribute given the current context. This means that OWL-based policies cannot be directly enforced into the system, but need to be further processed. By adopting an object-oriented terminology, OWL-based policies can be viewed as policy types: they define the actions that are allowed in a set of context types. In order to be enforced in the real world, policy types need to be transformed into policy objects that associate sets of actions with specific instantiated

contextual conditions. In the policy specification phase, administrators have to define aggregation and evaluation rules to enable effective enforcement and adaptation of OWL policies. For instance, in the meeting scenario an instantiation rule is needed to instantiate the current project attribute value included in the specification of the Colocated_Meeting_Actor class. The resource administrator could also define an aggregation rule to represent the "co-location" property as a relationship path based on the "location" property by means of variables.

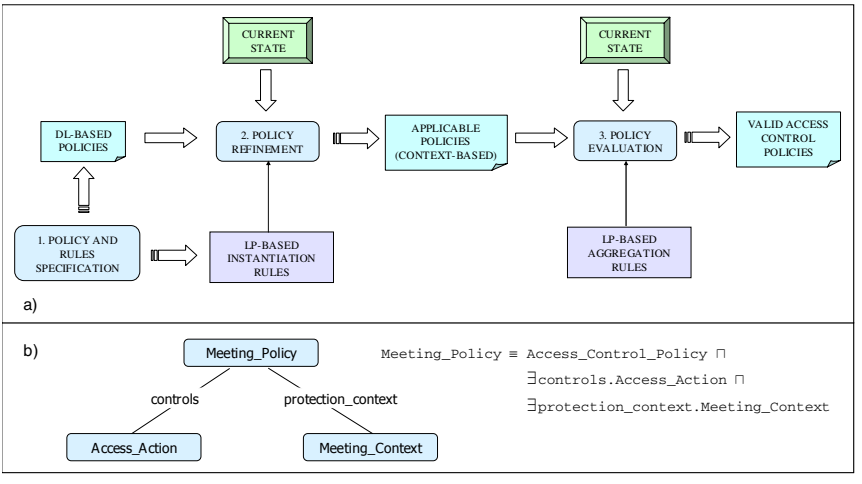


Fig. 2. The Context-Aware Policy Model and the DL-based meeting policy specification

In the *policy refinement* phase, OWL policies are instantiated by adapting them to the particular state of the world, in order to obtain the set of applicable policies. In the *policy evaluation* stage, the protection contexts of applicable policies are verified against the current state of context elements as measured by sensors to determine the set of currently active policies. Let us note that the context-aware transformation process comprising of policy refinement and evaluation may be triggered by any resource context change, such as a new user requesting to access the resource or a significant change in the resource state, e.g., its location.

It is worth noticing that our policy model adopts a combined approach to policy specification and reasoning. DL reasoning is exploited to perform static classification and conflict resolution of context and policy ontologies. LP reasoning is used to adapt the specification of OWL policies to the current state and allow their dynamic evaluation at access request time by means of appropriate rules. Adopting a combined approach allows us to benefit from the advantages of a pure ontology-based approach and those of a pure rule-based approach, both of which exhibit some limitations with respect to the definition and evaluation of policies and contexts [6, 7]. It is worth noting that our context model does not require the tight integration of the DL and the LP logical frameworks, which have well-known logical mismatches, but it is rather a combination of the two aiming at achieving more expressive description and reasoning capabilities about contexts and policies.

In the following subsections we focus on the policy refinement and evaluation phases which characterize our model and distinguish it from other state-of-the art related access control solutions [8, 9, 3].

3.2.1 Policy Refinement

Let us recall the meeting scenario to describe how policy refinement works. In the protection context of the meeting policy, shown before, the resource requestor property must belong to the Co-located_Meeting_Actor class that imposes that the resource requestor is co-located with the resource owner. Table 1 shows the definition of this context element, using a compact DL notation instead of OWL. Let us consider the restrictions applying to the properties is_currently_working_on and is_involved_in. These properties are restricted to a variable value, represented by the Current_Project class. This is an intrinsically variable value since the current project varies over time due to the changing activities of the resource owner and requestor, thus corresponding to different instances at different time instants.

Table 1. Co-located_Meeting_Actor class specification and instantiation and aggregation rules

Colocated Meeting Actor Specification	
Meeting_Actor ≡ ∃is_currently_working_on.Current_Project ⊓ ∃is_involved_in.Current_Project ⊓ ∃colocated_with.Resource_Owner	
Instantiation Rules to be applied in case of an ordinary scheduled meeting	
Current_Meeting_Rule	Scheduled_Calendar_Slot (?x) ∧ Meeting (?x) → Current_Meeting (?x)
Current_Project_Rule	Current_Meeting(?x) ∧ Project(?y) ∧ meeting_on_project(?x,?y) → Current_Project(?y)
Instantiation Rules to be applied in case of a meeting prolongation	
Current_Project_Rule-2	Actor(?y) ∧ Last_Current_Project(?x) ∧ is_currently_working_on(?y,?x) ∧ Scheduled_Calendar_Slot(?z) ∧ Idle(?z) → Current_Project(?x)
Current_Meeting_Rule-2	Scheduled_Calendar_Slot(?x) ∧ Idle(?x) ∧ Past_Calendar_Slot(?y) ∧ Meeting(?y) Current_Project(?z) ∧ meeting_on_project(?y,?z) → Current_Meeting(?y)
Aggregation Rule to determine co-location	
Colocation_Rule	Actor(? x) ∧ Actor(?y) ∧ SymbolicSpace(?z) ∧ located(?x,?z) ∧ located(?y,?z) → colocated_with(?x,?y)

The defined context instantiation rules are used to determine the correct instance of the current project class at access request time. In particular, let us consider the first couple of rules shown in Table 1. The first rule establishes that, if the user's calendar shows a meeting for the current time, then that meeting has to be considered the current meeting. The second rule states that the project discussed at the current meeting is the current project. Once the facts about the user's calendar are inserted into the refinement fact base, the first rule is triggered and the inferred current meeting instance is used as a new fact to trigger the second rule. Then, the protection context is instantiated by re-writing it with the inferred context element values. For instance, if SwapMe-Meeting is scheduled on the user calendar, and SwapMe-Project

is the corresponding project, then Current_Project is replaced by SwapMe-Project in the Colocated_Meeting_Actor specification. A new protection context is thus instantiated with the SwapMe-Project value and the corresponding policy generated with the instantiated protection context.

The combined adoption of OWL policies and LP rules enables policy adaptation when needed. For example, let us suppose that the meeting has gone beyond the allotted time. Given this state, the first group of rules cannot be applied because there are no valid facts in their head. Therefore, a new set of rules has to be defined during the definition phase to cover the situation of an extended meeting. In particular, the first rule determines the owner's current project on the basis of her past and current activities, independently from her calendar schedule. For instance, if the last instance of current project (determined at pre-defined intervals or at access request time) was the SwapMe-Project, if the calendar does not show any event for the current time, and if the actor is working on the SwapMe-Project, then the SwapMe-Project is still the current project instance. The second rules checks for the last and the current scheduling in the actor calendar. If there is no current event, and the last event was a meeting, and that meeting was about the current project (as determined with the first rule), then the last meeting is also the current one. In our example, the current meeting instance is the SwapMe-Meeting.

3.2.2 Policy Evaluation

We now describe the evaluation phase by using the same meeting scenario. When the current state of context elements, measured by sensors, is matched against the protection context of the meeting applicable policy, it is necessary to determine whether the protection context is currently in effect. During the evaluation phase the Co-located_Meeting_Actor definition of Table 1 is considered as well as the aggregation rule of Table 1 stating that if two actors are located in the same place (defined with the use of variables), they are co-located. Then, the resource owner's and the requestor's location are determined and inserted as facts into the evaluation fact base, which causes the execution of the co-location aggregation rule. Let us suppose that the requestor is co-located with the resource owner. In this case, a new fact is inferred that states that the resource requestor is co-located with the owner. This information is used to build the description of the current state of the world. In particular, an instance of the resource requestor element is created using the resource owner (which is known) as the value for the attribute co-location, and this instance of requestor is used in the protection context instance that describes the current state of the world. The created protection context instance is then compared with the protection context of the meeting policy by making use of ontology classification to recognize whether the former is an instance of the latter.

4 Related Work

Several research efforts have addressed the issue of access control in dynamic environments. We do not intend to provide a general survey of the state-of-the-art access control solutions in dynamic environments, but only to focus on the research that either integrates context-awareness and semantic technologies into access control

policy frameworks for pervasive environments or addresses access control issues in similar coalition application scenarios.

Considering context explicitly for access control is a very recent research direction with only few context-dependent policy model proposals. The importance of taking context into account for securing pervasive applications is particularly evident in [8] that allows policy designers to represent contexts through a new type of role called *environment role*. Environment roles capture relevant environmental conditions that are used for restricting and regulating user privileges. Permissions are assigned both to roles (both traditional and environmental ones) and role activation/deactivation mechanisms regulate the access to resources. Environmental roles are similar to our contexts in that they act as intermediaries between users and permissions. However, because environmental roles are statically defined in terms of attribute-constant value pairs their evaluation cannot provide support for policy adaptation as in our proposed semantic context-aware approach. In addition, differently from our approach, in [8] there is no integrated support for representing at a high level of abstraction and reasoning about environmental roles and policies.

By focusing on access control in spontaneous coalitions in pervasive environments, [3] proposes a delegation-based approach, where users participating to a communication session can delegate a set of their permissions to a temporary *session role*, in order to enable access to each other's resources. In particular, one end-point user assigns the session role to the entities he is willing to communicate with. Contextual information is used to define the conditions that must hold in the system in order for the assignment to take place, thus limiting the applicability scope of this process. Only a limited set of contextual information can be specified and no semantic technologies are exploited to represent nor the session role nor the delegation context constraint. In addition, security problems may arise whenever an entity delegated to play the session role leaves the communication session. In fact, unless the user explicitly states she is leaving the session, there is no way for the framework to be aware that the session role must be revoked for the departing user.

The importance of adopting a high level of abstraction for the specification of all security policy building elements (subjects, actions, context, etc..) is starting to emerge in well-known policy frameworks, such as KAoS and Rei [9]. KAoS and Rei represent, respectively, significant examples of DL-based and LP-based policy languages. In particular, KAoS uses OWL as the basis for representing and reasoning about policies within Web Services, Grid Computing, and multi-agent system platforms [10]. Contextual information is represented as ontologies and is used to constrain the applicability of policies. The KAoS approach, however, relying on pure OWL capabilities, encounters some difficulties with regard to the definition of certain kinds of policies, specifically those requiring the definition of variables. Rei adopts OWL-Lite to specify policies and can reason over any domain knowledge expressed in either RDF or OWL [11]. A policy basically consists of a list of rules expressed as OWL properties of the policy and a context represented in terms of ontologies that is used to restrict the policy's applicability. Though represented in OWL-Lite, Rei still allows the definition of variables that are used as placeholders as in Prolog. In this way, Rei overcomes one of the major limitations of the OWL language, and more generally of description logics. i.e., the inability to define variables. On the other hand, the choice of expressing Rei rules similarly to declarative logic programs

prevents it from exploiting the full potential of the OWL language. In particular, the Rei engine is able to reason about domain-specific knowledge, but not about policy specification. Our policy model shares some commonalities with regard to context/policy representation with both KAoS and Rei, but differs in how it deals with context. Our approach considers context as the primary basis that allows one to deduce which policies apply to a subject acting in the system whereas KAoS and Rei, similarly to traditional approaches, exploit context to build filtering mechanisms for policy applicability.

5 Conclusions and Future Work

The dynamicity and heterogeneity of pervasive scenarios introduce new access control challenges. A paradigm shift in policy models is needed to move focus from the identity/role of the principal to the context that the principal is operating in. We propose a semantic context-aware policy model, which treats context as a first-class principle for policy specification and adopts a hybrid approach to policy definition based on DL ontologies and LP rules. We are currently working on implementing a prototype for the meeting scenario using OWL to specify ontologies and SWRL to encode rules. For this implementation, we are using Pellet [www.mindswap.org/2003/pellet/] to reason about ontologies and Jess [herzberg.ca.sandia.gov/jess/] for forward-chained reasoning about rules, both accessed through a Java interface (via Jena [jena.sourceforge.net/] with Pellet). We are also working on the design of a deployment model that includes different components in charge of monitoring contexts, installing policies into the system, performing policy refinement and evaluation, and enforcing policies. Future work will include providing alternative implementations of the model using different languages, such as N3Logic [http://www.w3.org/DesignIssues/Notation3.html], which provides a uniform notation for ontology and rule specification, and the cwm reasoner [http://www.w3.org/2000/10/swap/doc/cwm.html]. We also plan to further develop application scenarios in order to analyse the usability and effectiveness of our semantic context-aware model.

References

[1] Dey, A., Abowd, G., and Salber: D.. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16:97-166, 2001.

[2] Sandu, R., et al. : “Role based access control models”, IEEE Computer, Vol.29, No.2, February (1996).

[3] Liscano, R. and Wang, K.: “A SIP-based Architecture model for Contextual Coalition Access Control for Ubiquitous Computing”, In: Proceedings of the Second Annual Conference on Mobile and Ubiquitous Systems (MobiQuitous ’05). IEEE Computer Society Press (2005).

[4] Georgiadis, C.K., et al.: “Flexible Team-Based Access Control Using Contexts”, In: Proc. of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001), May 3-4, Chantilly, Virginia, USA. ACM (2001).

[5] Gong, L.: "Inside Java 2 Platform Security", Addison Wesley, 1999.
[6] Lassila, O. and Khushraj: D., "Contextualizing Applications via Semantic Middleware", In: Proc. of the Second Annual Conference on Mobile and Ubiquitous Systems (MobiQuitous '05). IEEE Computer Society Press (2005).
[7] Toninelli, A., Kagal, L., Bradshaw, J.M., and Montanari, R.: "Rule-based and Ontology-based Policies: Toward a Hybrid Approach to Control Agents in Pervasive Environments." In: Proc. of the Semantic Web and Policy Workshop (SWPW), in conj. with ISWC 2005, Galway, Ireland, Nov. 7 (2005).
[8] Covington, M.J., et al.: "Securing Context-Aware Applications Using Environmental Roles", In: Proc. of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001), May 3-4, Chantilly, Virginia, USA. ACM (2001).
[9] Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: "Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder", In: Proc. of the Second International Semantic Web Conference (ISWC2003), LNCS, Vol. 2870. Springer-Verlag, Berlin, pp. 419-437, Sanibel Island, Florida, USA, October 2003.
[10] Uszok, A., et al.: "KAoS policy management for semantic web services". IEEE Intelligent Systems, 19(4), p. 32-41, 2004.
[11] Kagal, L., Finin, T., Joshi, A.: "A Policy Language for Pervasive Computing Environment" In: Proc. of IEEE Fourth International Workshop on Policy (Policy 2003). Lake Como, Italy, pp. 63-76, IEEE Computer Society Press 4-6 June 2003.