Community Based Policy Management for Smart Spaces

Kevin Chekov Feeney, Dave Lewis

Knowledge and Data Engineering Group, Trinity College Dublin
{kefeeney, dave.lewis}@cs.tcd.ie

Abstract. Access control in Smart Spaces requires flexible, dynamic and powerful management approaches. Policy based management has the potential to empower users and administrators in this domain. Rigid organisational models with pre-determined policy rules limits the possibilities for users' interactions with the space and between each other. This paper outlines a community based policy management approach, which allows administrators to set boundaries within which users can manage the resources in the space themselves. It examines this model using the test case of a Smart Space in a university and finally describes the initial implementations of the community based policy management framework.

1. Introduction

Smart Spaces require access control mechanisms to allow users and administrators to access resources and services in a controlled way. However, much of the research into Smart Space management has focused on providing a limited number of wireless services to a clearly defined user base in a specific environment. For example, the provision of location-aware multimedia content to visitors to museums [1], or providing a small number of location-aware services to students on a college campus [2]. These projects often adopt a very simple user model, where every user has the same rights and privileges as the others with respect to the Smart Space services, and they assume a static, pre-defined set of Smart Space services which only apply to a limited number of devices. This allows them to make assumptions about access control and security, which do not extend to envisaged real world Smart Spaces where devices and services will be able to dynamically interact to achieve user goals.

When attempting to devise general access control models for real-world Smart Space management systems, there is a need to accommodate a much greater diversity in terms of the users, devices and services available. This could be achieved by using traditional Access Control Lists (ACL's), whereby each resource is associated with a particular ACL and only users included in the ACL have access to that resource. However, the dynamic and fluid nature of projected Smart Spaces render this approach impractical as the number and complexity of the ACL's would quickly exceed the capacity of human administrators to manage them. Policy Based Management (PBM) provides several advantages over traditional ACL's since they have the potential to be much more expressive and powerful, In addition there are a

number of very good reasons specific to Smart Spaces, for using PBM in any Smart Space access control model.

2. Policy Based Management of Smart Spaces

PBM rests on the assumption that a common set of rules can be applied to sets of entities in the system, rather than merely to individual entities. Thus, policy languages include constructs to group entities together. This facilitates management by allowing the definition of rules that apply to sets of entities rather than to individuals. Policy rules are divided into authorisation policies for access control management and obligation policies for resource management [3].

In Smart Spaces, a user's access rights to resources and services are not necessarily static. Access rights can vary due to context. For example, a user may have different rights depending on what other users are present in the space. This dependence of rights on context presents a difficult problem for traditional access control mechanisms. However, when using policy languages and Role Based Access Control (RBAC) [4], these contextual changes can be modeled as changes in memberships of the roles or other grouping abstractions that form the basis of the policy language. The roles available to a particular user can change according to the context. Therefore, as long as we can manage this type of role variability, the access control system does not need to be changed in any way to take account of context - a significant advantage.

Furthermore, Smart Spaces present particular problems to traditional authentication systems. Smart Spaces can be composed of a variety of services, some of which will not be particularly sensitive from a security point of view, while others may require high levels of confidence in the authentication of users. Ideally we would like to allow users to authenticate themselves to the Smart Space only to the required level for those services which they will use during their presence in the Smart Space. By associating different types of authentication with different roles and allowing users to choose which roles to activate upon entering the Smart Space, we can provide a seamless, integrated means by which a user can authenticate herself to the Smart Space to the minimum required level. For example, a typical user might be able to authenticate herself to a space by means of an active badge [5], but if the user wants to activate her 'administrator' role, a password or biometric identification mechanism may be required. Thus variable levels of authentication can be associated with different roles in a seamless manner.

The Gaia project [6] at the university of Illinois has adopted a powerful model for security and access control using roles, policies and different authentication methods with variable confidence levels among other measures. Gaia uses credentials, similar to Kerberos tickets, as a basis for authentication and access control. In Gaia, all resources in the system are associated with policies. Users present credentials to the system which are compared with policies in order to validate particular resources. Credentials are closely linked to roles. Users can activate any subset of their valid roles and receive credentials associated with these roles, which then allow then them to access resources.

However, current approaches to PBM have serious limitations when they are applied to Smart Spaces. The most important problem in most PBM approaches is that the role based organisational model must be defined in advance in a complex process of requirements engineering. A class of specialist administrators subsequently applies modifications to the model. The goal of exhaustively describing the operations of an organisation through a static set of roles and policies that are applied to these roles is problematic even when dealing with relatively structured and static domains. When applied to the dynamic and unstructured realm of Smart Spaces, this approach quickly becomes a serious limitation on interactions in the space. For example, the management model adopted in the Gaia project requires that the space administrators define a set of modes for the space which specify the roles of the participants and their rights with respect to the available services. Thus, users are effectively prevented from interacting with the space in ways that have not been envisioned in advance and programmed by the administrators. It would be much preferable to adopt a management model which allowed administrators to define rules to bound the behaviour of users while allowing them to develop ways of interacting within the space through the process of progressive grounding [7].

This paper examines the potential for applying Community-based Policy Management (CPM) [8] to the management of Smart Spaces. CPM enables the self-management of spaces by the users through the dynamic creation of communities, which delegate authority over the resources in the space among themselves and thereby facilitate a much greater flexibility and dynamism in the interactions between users and the space while still allowing space administrators to enforce rules which bound the behaviour of users.

3. **Community** based **model.**

In this section, we provide the basic outlines of a Community Policy Based Management framework, where an organization's structure is modeled as a set of interrelated communities. Authority over particular resources is distributed throughout the organization using delegation between communities. This model is intended to reflect the way that fluid communities work in practice. Rather than creating the structure through a centralized modeling process, the framework enables a policy based approach to be applied to the management of the processes of sub-division, delegation of authority and incorporation through which these communities dynamically define their own structures and operational rules.

In our framework the community is specified as:

- A set of membership rules, The membership rules are similar to the role activation constraints of constrained RBAC [9] or OASIS [10], in that they allow us to specify the conditions that must be met before an individual is admitted into the community.
- A set of sub-communities. Sub-groups within the overall community are themselves modelled as communities, their membership being a subset of the parent community.

- The set of policies that apply to the community. This is all of the policy rules in the system which have the community as their subject.
- The set of resources that the community has authority over. A resource can be composed of other resources. Having authority over a resource implies that the community has an authorisation to perform some action on the resource and also to delegate a portion of its authority over the resource to a sub-community. Resources can either be owned by the community, or delegated by a parent community.

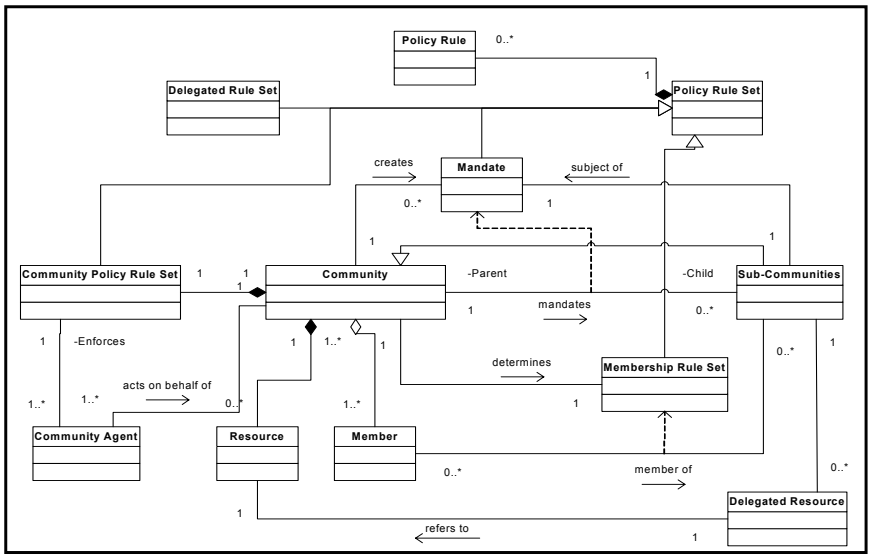

Fig. 1. Community Meta-Model

At its simplest, any organization is modeled as a simple, undifferentiated community. The modeling of specific groups within the community is achieved through the creation of mandated sub-communities or child-communities, each made up of a subset of the membership of the root community. Sub-communities can themselves create their own sub-communities. The creating community becomes the parent community of the sub-community. Each sub-community has exactly one parent to ensure a simple hierarchical relationship, or partial order, between communities. Any community can create a sub-community by defining membership rules for it. The sub-community is described as mandated because the parent community may create policy rules whose subject is the sub-community. That is to say that the parent community may define what the sub-community is authorized to do (by defining authorization policies whose subject is the sub-community) and what it is obliged to do (similarly, by defining obligation policies). The set of policies that have been defined by the parent community and whose subject is the sub-community is the mandate of the sub-community.

3.2 Delegation of authority and rights

Delegation of authority and rights is fundamental to the community policy framework. Any community can delegate authority over any subset of its resources to its sub-communities. If a community has authority over a resource, this means that it can author policies whose target is that resource. When this authority is delegated to a sub-community the sub-community can author policies whose target is that resource. This concept of delegation differs from the concept of delegation in Ponder. Ponder's delegation model allows individuals to nominate others who are trusted to act on their behalf to perform certain tasks, independent of the delegated individual's position in the organizational structure. We describe this type of delegation as delegation of identity, as the delegated individual acts as if she was the delegating individual, by effectively assuming their identity for the enactment of particular tasks. Delegation of identity is an important feature for any PBM system, as it is a common practice in all organizations. However, it is a different concept to the delegation of authority between communities and the two concepts are entirely compatible within a PBM system.

Practically, delegation of authority amounts to entering a reference to that resource in the set of delegated resources of the sub-community, as shown in figure 2. Similarly a policy that applies to a community can be delegated to its sub-community by inserting the policy rule into the sub-community's set of policies. Negative policies are not explicitly delegated since if a negative policy applies to a community, then it implicitly applies to its sub-communities. A community can't permit its sub-community to do something that it is not itself permitted to do. Similarly, constraints that apply to policies, whether they be meta-policies limiting the form of the policy rule or constraints on the conditions under which a policy is valid, are implicitly propagated from a community to its sub-communities.

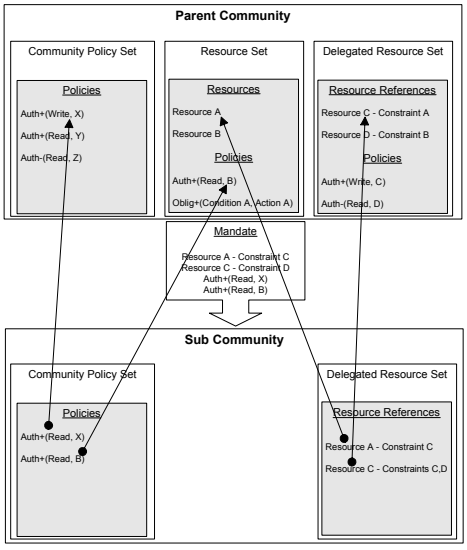

Fig. 2. Delegation to sub-community

3.3 Resource Semantics

In the CPM framework, resources are associated with an 'authority tree', which is a partial order of actions that can be taken on the resource. If a community possesses authority to take a particular action in the tree, it also implicitly possesses the authority to take any action beneath that in the tree on the same target (or on a subset

of that target) and can delegate any of these actions to a sub-community. This hierarchical tree provides a partial order on actions, according to the implies relationship, described in [shen]. Authority to perform a particular action implies authority to take sub-actions further down the tree.

Target resources are also organized in a hierarchical domain structure. This enables the community management system to ensure that all delegations are valid. If a community possesses a right *Auth+{ActionA, TargetA}* then a delegated right *Auth+{ActionB, TargetB}* is validated by ensuring that the right to take *ActionA* implies the right to take *ActionB* with respect to *TargetA* and that *TargetA* is a parent of or is equal to *TargetB* in the resource tree.

3.4 Decision making and community agents

What do we mean when we say that the community can define policies and that the community is the subject of policy rules? The community is not a single entity and can be made up of many individuals with independent motivations, who can not be trusted to always carry out the community's will. Thus, we introduce the concepts of community decision-making mechanisms and community agents into our model. Community agents, whether human or automated, carry out actions on behalf of the community. A community member becomes a community agent when a decision has been made by the community to carry out a specific action. Decision making mechanisms define the process through which a community reaches a decision. A community may decide that some actions can be taken on behalf of the community by any member of the community, while other actions may require agreement by several members, or even a majority vote, before they can be taken on behalf of the community. Once a decision has been made the community issues a certificate to the community agent which permits the agent to perform the action on behalf of the community.

3.5 Internal Privacy and Hierarchies

Authority over resources may be delegated repeatedly from communities to their sub-communities and each may apply policy rules to the resource. However, each community delegates only to its immediate sub-communities and allows them to subdivide the authority between their own sub-communities. Thus communities are only aware of their parent community and their immediate children. This allows communities to keep their internal organisation private from the rest of the broader community. When a community delegates a policy to a sub-community, only an agent of that sub-community can act upon the policy.

Although they can be self-managing, there are also cases when communities are managed by other communities. For example, users of various services may be considered to be part of a top-level community, yet those who have direct involvement with its development take decisions on behalf of the broader community. This can be accommodated in our model by specifying a particular sub-community as a control community. A control community acts on behalf of its parent community.

3.6 Interpreting Policy Hierarchically

Through the process of communities sub-dividing and delegating authority to their sub-communities, a hierarchical tree of communities is built up within the overall community and this amounts to a model of the distribution of authority over each resource managed by the community. This model is used in the delegation process, to ensure that any authority delegated is a subset of the authority possessed. However, it also provides us with a means of enforcing policies in a hierarchical way to automatically resolve policy conflicts. Even when a parent community delegates authority over a particular resource to a sub-community, the parent may still author policies whose target is the resource. This creates potential policy conflicts on particular resources due to conflicting policies being defined by communities and their sub-communities (which may occur several steps of delegation away). To get around this problem, policy is hierarchically enforced. Thus, policies authored by the community that owns the resource have precedence over policies authored further down the hierarchy. As the hierarchy is based upon authority with respect to the resource, this policy precedence is correct by definition. This is an important feature since it allows us to define community-wide policies at the most general levels, policies that will continue to be enforced irrespective of any policies that are applied in sub-communities. For example, we can impose community-wide security guarantees on all resources in the top-level community while leaving the sub-communities free to define extra policies in their specific areas of responsibility, without the risk that these policies may inadvertently violate the security guarantees.

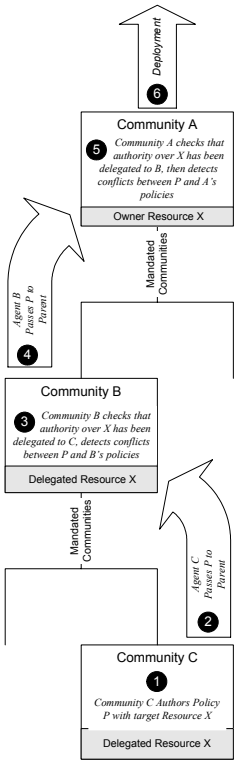

Fig. 3. Hierarchical Policy Enforcement

When a new policy is authored anywhere in the community structure (step 1 in figure 4), it passes up through its parent communities (step 2) until it reaches the owning community of the target resource. At each stage the parent automatically checks to ensure that the new policy rule does not conflict with its existing policies for the target resource (step 3) before issuing a community agent certificate and passing it on to its parent community (step 4). When the rule reaches the owning community, it is deployed to the target resource (step 6). If a policy conflict is detected before deployment, the new policy rule is either rejected or, if possible, automatically rewritten to limit the target domain in order to avoid the conflict. A rejected policy is returned to the authoring community with an indication of the level on which the conflict was detected. The authoring community can then choose to manually rewrite the policy rule to avoid the conflict, or can attempt to negotiate a change in the policy rule at the level where the conflict was detected (which it is by definition a subset of). As policy conflicts can be the result of genuine conflicts

between communities, caused by conflicting demands and resource competition, there is no way to avoid all cases of manual re-negotiation of policy rules. In these cases, we need to know at what level of the organisation the conflict must be resolved and the community policy achieves this by identifying the nearest common ancestor of the conflicting communities. This hierarchical enforcement of policy depends upon a conflict detection mechanism. The discussion of conflict detection is beyond the scope of this paper, for a comparison of various approaches see [11,12]).

4. Case Study

We demonstrate the operation of the CPM system in the context of Smart Space management through the construction of a case study. This case study involves a Smart Space in a University, which is intended to be used as a general space and which may be required to host lectures, presentations, meetings or as an information space for researchers and students. The space is equipped with several resources, which are managed by the CPM system, namely an electronic whiteboard, a projector, a network connection with limited bandwidth and environmental controls, which set temperature, ventilation and lighting. The case study looks at the use of the space for a period of time when the space is firstly used as a free-access work space, with all the facilities available to whoever wants to use them, then as a lecture and presentation space.

4.1 Administration

The space administrators construct a very basic community structure for the management of the space. This top-level community includes all those users who are eligible to use the space and all of the space administrators. The qualification rule for this community is that the user is registered with the university as a student, staff member or guest. The top-level community has a control sub-community to which the decision making is delegated, this community is composed of the space administrators. In order to address the managed resources, the administrators of the space construct a hierarchical domain map of the resources in the space. The community structure and the community policies are themselves treated as managed resources. Each resource is associated with a particular authority tree [13], as shown in figure 4. Authority to perform a particular action in the tree implies authority to perform all of its sub-actions and to delegate that authority to its sub-communities. Any policies authored by this community are applied as if they were authored by the root community and thus have precedence over any policies authored by other sub-communities. This allows the administrators to apply bounds to the utilization of resources. For example, the administrator community may specify a maximum network bandwidth that can be used for the entire space and can specify minimum and maximum temperatures that can be set using the heating controls.

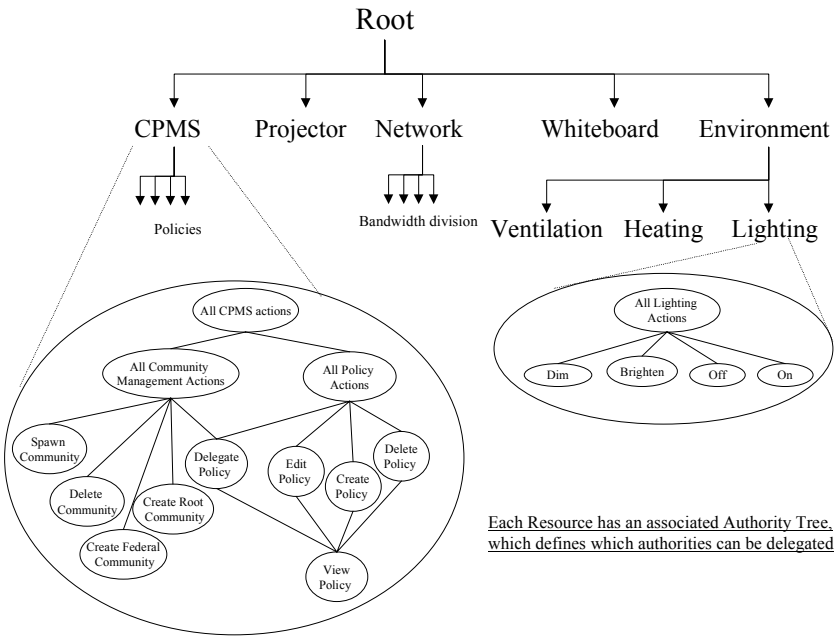

Fig. 4. Resource Tree for University Smart Learning Space

The administrators then define another sub-community, that of current users (those users currently in the space) to which it delegates authority over all resources. This community is then free to take all actions which do not contradict the policies specified by the parent community. This community has access to a set of decision making methods, with Majority vote being the default decision making method. Finally, the administrators define a further sub-community of the current users, namely the current lecture group community. This community is a control community of the current users community. The qualification rule for this community specifies that the space must have been booked for this specific time and that the user is a member of the group that booked the space. This allows specific groups to book the space and take control over the resources in it, regardless of whoever else might be present. Unless circumstances reveal a flaw in their policies, the space is henceforth entirely self-managed by its users – through the PBM framework.

4.2 Usage

At the start of the day, a single researcher enters the space and, as the only member of the current users community, gains control over all of the resources. She sets the heat, lighting and ventilation to her tastes and decides to settle down to watch television on the projector screen. After some time, two students enter the space in order to run through some proofs on the whiteboard. The resources are now

controlled by majority vote of all 3 current users. They negotiate the environmental settings between them so that they are all happy. They then decide to form two new sub-communities to manage their different tasks. They delegate control over the whiteboard to the students and the projector to the researcher, as well as granting her the majority of the bandwidth. This means that, although the resources are ultimately controlled by the set of total users, the sub-communities can manage the delegated resources without referring to those who are not involved in the relevant task (fig. 5).

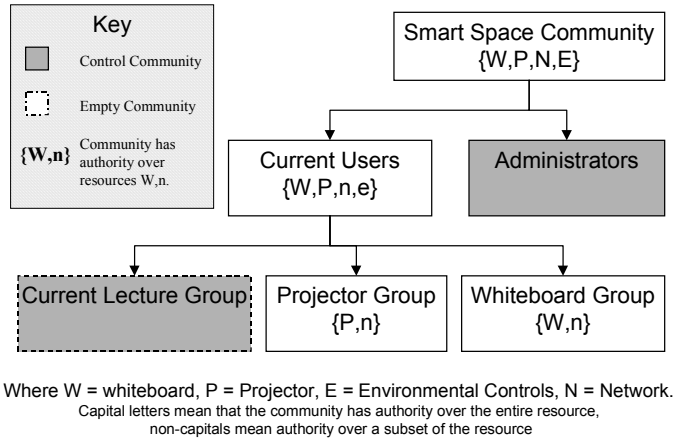

Fig. 5. Community Structure 1

As time passes, several other students enter the space and join one of the two groups. The space has been reserved for a series of presentations at 10 am, and when this time arrives the users who have booked the space gain are made into active members of the lecture group community and gain control over the space's resources. As soon as the lecture group has assembled, they delete the existing task based communities and re-allocate control over the resources. They first create a sub-community of presenters and delegate control over the projector, bandwidth and the lighting to them. This community creates a control sub-community for the current presenter, with a qualifying rule of whoever is standing closest to the podium. They also create a policy allowing anybody to add information to the whiteboard. The

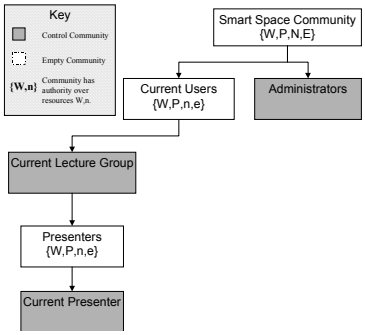

Fig. 6. Community Structure 2

presentations proceed and as each presenter comes to the podium, they automatically gain control of the lighting, projector and network bandwidth, while the rest of the participants can add comments to the whiteboard.

The advantage of this management approach is that the space administrators can define a very minimal set of communities and rules and then allow groups of users to refine them through the delegation of resources and the creation of sub-communities. This puts few constraints on the applications and interactions between the users and the space while still ensuring that crucial constraints are not broken. The system assumes that all users have access to some form of input/output device in the space which enables them to participate in community decisions, however in practice users could be represented by user-agents operating through their PDA's or according to user preferences. So, for example, environmental settings can be agreed by comparing user preferences without any active input from users.

5. Implementation

Although it is not directly implementable in any existing PBM system, since they lack community constructs with the required semantics, the community policy framework is not tied to any specific policy language and uses broadly the same policy concepts as other existing PBM systems. Therefore, the approach of mapping the community based model to an existing PBM framework was thought prudent rather than devising another Policy language and architecture. The Ponder framework and its suite of tools, along with the source code, are freely available to download on the Internet. Ponder uses a declarative object-oriented language which is specifically designed for simple policy specification - particularly suitable for this domain where policy rules are not authored by experts. Furthermore, Ponder provides a range of grouping constructs and the fact that Ponder policies are themselves managed resources allows a great degree of flexibility in designing an administrative model. Therefore, Ponder was used in our initial experiments as an underlying policy framework.

The community structure is defined in XML and interpreted with a custom software application, written in the Java language using the JAXML, JNDI and Ponder libraries. It validates each delegation of authority to ensure that the delegated authority is possessed by the parent community. This is currently achieved by using a simplified hierarchy of actions and target resources that are organized in a hierarchical directory structure. Any constraints placed upon policies by parent communities are also checked. Currently, the software only detects those conflicts that Lupu describes as modal [12], such as the existence of positive and negative authorization policies with the same subjects and targets. Once all of these checks are passed, a set of domains, policies and roles to implement the rules of the community structure is created in an LDAP directory, corresponding to Ponder language constructs. The membership rules are evaluated and Ponder user objects are created and assigned domains. Communities are mapped to Ponder domains and a set of policies that dictates the rights and obligations of the members of the domain. The hierarchy of the communities is reflected in the resulting Ponder domain hierarchy. In

the original version each member of a community acts as an agent of the community, without decision-making mechanisms, as this concept is not supported by the software framework. Furthermore, the concept of hierarchical enforcement of policy does not fit easily into Ponder's deployment framework, and this was not implemented in the original version. Nevertheless, the Ponder mapping of the community structure is capable of implementing many of the features of the community model. In particular, the delegation of authority through the community is validated by the software and mapped into a simple set of domains, roles, policies and meta-policies in Ponder.

Having demonstrated the feasibility of this approach using the Ponder language as an underlying policy framework, we have used it as a basis from which to develop a simple custom policy language that can provide direct support for the community abstraction and the decision-making methods required by the model. The fact that we were able to utilize this simple policy language in place of Ponder demonstrated the general applicability of our model. As long as the underlying policy framework allows us to identify policy conflicts and we have semantic information available about the managed resources to construct an authority tree, the community framework is neutral to the actual language used. This raises the possibility of using the CPMS as a means of managing organizational structures and authority in an environment where several different policy languages may be used.

5 CONCLUSIONS

This paper has presented a community-based model for the management of access control in Smart Spaces. This model builds upon previous work in the field of Policy Based Management, but simplifies the specification of the organization by using a single grouping construct, the community, to model the human organization. It introduces a concept of delegation whereby authority and rights are delegated to dynamically build a model of the organization. Communities retain all of the advantages of specifying policies for roles or domains: policy decisions are made on the basis of the communities that a person belongs to, rather than their identity and policy can be composed through conjunction of the communities to which that an individual belongs.

However, communities also extend the standard role-model in several ways. In particular, communities provide a means of incorporating group context and decision making; they integrate administration into the model and allow policy rules to be interpreted with respect to the overall structure of authority in the organization. They facilitate the self-modeling of groupings within the community, autonomous control of shared resources and allow for privacy of different parts of the structure. A hierarchy of communities provides a means for modeling the organization in a much more succinct and intuitive way than by using roles with a variety of different relationships between them.

A primary strength of this framework is that it does not require a detailed process of requirements engineering to create a model of the organization. We can create the most basic structure and allow the detailed divisions of responsibility and the

organizational groupings to evolve in an organic manner. Thus, we can introduce a PBM system by merely modeling the entire organization as a single community, with authority over the full set of resources managed by the system. As the needs arise, we can create sub-communities and delegate to them responsibility for specific resources. The analysis of policy conflicts can be used to signal structural problems in our model and in the underlying real-world community, thus providing constant feedback to refine the model.

References

1. Semper, R., Spasojevic, M.: "Exploratorium - The Electronic Guidebook: Using Portable Devices and a Wireless Web-based Network to Extend the Museum Experience - Overview and Findings to Date", HP Labs. Paper presented at Museums and the Web Conference, April 2002
2. W.G. Griswold, R. Boyer, S.W. Brown, T.M. Truong, E. Bhasket, R.Jay,R.B. Shapiro: ActiveCampus: Sustaining Educational Communities through Mobile Technology University of California, San Diego,Department of Computer Science and Engineering, Technical Report (2002)
3. Sloman, M., Lupu, E.: Security and Management Policy Specification, IEEE Network, Vol. 16, no. 2, (2002) 10-19
4. Sandhu, R.S. et al., Role Based Access Control Models, IEEE Computer, vol. 29, no.2, (1996) 38-47
5. Sandhu, R.S. et al., Role Based Access Control Models, IEEE Computer, vol. 29, no.2, (1996) 38-47
6. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System, ACM Transactions on Information Systems, Vol. 10, No. 1, (1992) 91-102
7. Barrett, R.: People and Policies: Transforming the Human-Computer Partnership, 5th IEEE International Workshop on Policies and Distributed Systems and Networks, IEEE (2004) 23-35
8. Feeney, K., Lewis, D., Wade, V.: Policy-based Management for Internet Communities, 5th IEEE International Workshop on Policies and Distributed Systems and Networks, IEEE (2004) 23-35
9. Sandhu, R. et al.: The ARBAC97 model for role-based administration of roles: Preliminary description and outline, Proceedings of 2nd ACM Workshop on Role-Based Access Control, (1997)
10. Hine, J., Yao, W., Bacon, J. and Moody, K.: An Architecture for Distributed OASIS Services, Proceedings of Middleware 2000, New York, USA, Lecture Notes in Computer Science, Springer-Verlag, (2000). 107-123.
11. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R.,Suri1, N., Uszok, A.: Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei,and Ponder, Proceedings of 2nd International Semantic Web Conference (ISWC2003) (2003)
12. Lupu, E.C, Sloman, M.: Conflicts in Policy-Based Distributed Systems Management, IEEE Transactions on software engineering, vol. 25, no. 6, (1999) 852-69.
13. Shen, H., Dewan, P.: Access Control for Collaborative Environments, Proceedings of the ACM Computer- Supported Cooperative Work Conference, (1992) 51-58