

Handling Dynamic Organizational Change with Community-Based Policy Management

Kevin Feeney, David Lewis, Vincent Wade
Knowledge and Data Engineering Group, Trinity College Dublin
{Kevin.Feeney|Dave.Lewis|Vincent.Wade}@cs.tcd.ie

Abstract

Policy-based management (PBM) aims to provide flexibility in the management of resources so as to readily reflect changing business goals. However, as organizations increasingly use electronic means for more of their core business operations, the ability to ensure that policies accurately reflect the operation of an organization becomes more challenging. This paper presents a critique of organizational modeling abstractions used in existing policy and access rule schemes. It then uses a detailed case study to explain how Community Based Policy Management can provide a more flexible approach to managing organizational change than the organizational abstractions currently proposed for PBM, in particular in comparison to roles.

1. Introduction

The modeling of organizations and their processes and business rules underwent a transformation in the course of the twentieth century. Taylorism [taylor], which concentrated on implementing systems to allow senior management to define precise operational rules governing all aspects of the production process, gave way to Human Resources Management (HRM) which recognized that the standardization and predictability aimed for in Taylorism could easily degenerate into rigidity and defensive behavior [thompson]. HRM and its offshoots attempted to factor in notions such as autonomy, decentralization and team-working and channel these aspects towards achieving the high level goals of the organization. The emergence of newer organizational paradigms, such as value chains, virtual organizations and Internet communities has further emphasized the need to model organizations as flexible and dynamic entities with the decentralized and autonomous management structures proposed in this project.

The emergence of Policy Based Management technology, which essentially aims to provide automatic translation between an organization's high level business goals and the configuration of its IT infrastructure has brought the problems of creating adequate organizational models into sharp focus. Current approaches to policy engineering generally rely upon a role based model [sandhu][lupu] that requires a detailed requirements engineering phase in order to precisely define the structure of roles and the rules applied to them within the organization. This approach is analogous to Taylorism in the electronic age and suffers similarly from excessive rigidity, difficult and expensive requirements modeling and a lack of dynamism [feeney07]. In order to circumvent these problems, we have developed a Community Based Policy Management (CBPM) model which supports much greater levels of flexibility, dynamism, decentralization and autonomy in management, making it particularly suitable for the management of new, less rigid organizational forms. Initial validation has been through integration with a collaborative web portal for the open source community [feeney04], a location-aware instant messaging application in a ubiquitous computing scenario [lewis04c][feeney05] and providing dynamic spectrum access policies to a cognitive radio systems [lewis06]. However, these integration scenarios furnish us with little evidence that CBPM offers a qualitative improvement over existing PBM schemes that use roles as the central organizational modeling abstraction. Experimental evidence for these claims is difficult to establish, without observing the operation of a large number of policy authors in rapidly evolving organizations. Due to the limited deployment of PBM systems in real-world industrial applications, there are few such suitable subjects and, by definition, these are busy people who are difficult to co-opt into academic studies. Instead in this paper, therefore, we aim to demonstrate these features of the CBPM scheme through a carefully constructed case study based on an existing organization's model.

2. State of the Art

Policy-Based Management systems allow rules to be specified that apply to classes of entities rather than individuals. These classes might serve as abstractions to group similar network devices together [boros], or they might group individuals together on the basis of them having similar roles in an organization [sandhu00][lupu] or they might divide organizations into management domains [sloman94a]. These grouping abstractions enable much more efficient management. For example, it is very convenient for IT managers to be able to specify rules that apply to all databases or all programmers rather than having to configure rules for each database and each programmer individually. The use of grouping abstractions is not unique to policy systems, and is common to many types of traditional access control management system. For example, discretionary access control models, commonly used by the Unix family of operating systems to control access to file system objects, allow permissions to be specified for groups as well as for individual users [doeppner]. Firewall configuration languages allow rules to be applied to classes of traffic. However, modern PBM systems aim to extend the scope of the grouping abstractions, enabling rules to be specified to classes of entities, classes which apply across heterogeneous systems and networks and are applicable and meaningful on multiple managed information resources [sloman94b]. For this reason, PBM requires powerful, flexible and expressive grouping abstractions, abstractions which may be more general than the class relationships commonly used in computer science.

Roles are the most common abstractions used to model subjects in PBM systems. The role abstraction has been standardised in the NIST Role Based Access Control (RBAC) model [sandhu00]. RBAC has its roots in research into access control and security and is thereby focused on providing formal models which are amenable to mathematical reasoning and allow particular security properties of the model to be formally proved. This places inherent limitations upon the expressiveness of the models produced, since the inclusion of a class of constraints, for example temporal constraints as in Temporal RBAC [bertino], requires the constraints to be integrated into the formal model – a difficult and labor intensive research task. Models such as Context-Team Based Access Control [georgiadis] and Organization-Based Access Control (OrBAC) [el-Kalam] extend the expressiveness of access control systems through the integration of contexts into the models – although this comes at the

expense of the ease with which security properties of the system can be formally proved. OrBAC further extends this expressiveness by adding negative authorizations, obligations and recommendations.

From our point of view, where we are interested primarily in the organizational model embodied in these access control models, we can say that none of them provides an integrated view of the organization in functional or structural terms. For a start, none of the models incorporates a description of how the model itself is administered. This is either specified as a distinct administrative hierarchy, as in Administrative RBAC [sandhu99], or partially related to the user model through the organizational scope, as in Administrative OrBAC [cuppens], or left undefined and assumed to be carried out by the security officer. Furthermore, none of the models capture a great deal about the function of the system and how that relates to the functions of the constituent parts. Although Team Based Access Control (TBAC) [thomas], Coalition Based Access Control (CBAC) [cohen] and OrBAC do provide support for structural grouping abstractions which can also have a functional significance, these abstractions are largely modeled in isolation from one another. Although OrBAC does contain the *sub_organization* relation for relating organisations together, the semantics of this relationship must be defined by the administrator – it has no inherent meaning. Finally, none of the models define abstractions which allow us to relate managed objects to one another or which allow us to relate users, or groups of users, to groups of objects.

The Ponder PBM framework [damianou] offers one of the richest set of grouping abstractions demonstrated to date. Ponder is extremely expressive and applicable to a large subset of the typical requirements of policy based management systems. It provides a wealth of grouping abstractions, many of which can group together virtually any of the structural units of the language into administrative units. We can use Ponder to construct a rich structural model of the organization, made up of management structures, groups of policies and roles. Role relationships provide a means of relating some of the structural elements together, while the nesting of management structures provides another means of relating the structural elements together into a model of the organization – a bottom-up and top-down model respectively. Meta-policies (policies which apply to policy objects) allow a precise definition of administrative authority, while delegation allows a certain measure of decentralization of administration. The roles responsible for the system's administration can be integrated into the

general organizational model, providing a more complete model of the organization.

However, despite and because of Ponder's flexibility, it provides little in the way of answers to the question of how we can best integrate a policy based management system with a human organization. Ponder allows us to create virtually any mapping between the human organization and authority vis a vis the policy system. The Ponder domain construct is the source of this flexibility. By modeling every Ponder entity as an object, and modeling domains as hierarchical sets of object references and by allowing objects to be referenced by multiple domains and by allowing arbitrary domain scope expressions to be the targets and/or subjects of policy rules, we gain a great deal of flexibility, but we also push many of the problems of organizational modeling beyond the scope of the system. Much of the organizational model is dependent on how we populate our domains. For example, the choice of which domains newly created objects are added to – including all of Ponder's structural elements – has a significant bearing on how the organization works. The use of roles is at the discretion of the policy author and policies can be specified for arbitrary sets of subjects, as well as for roles. Although meta-policies can be authored to limit the form of policies, this once more pushes the problem of organizational modeling onto the administrators. Similarly, the targets of policies can contain arbitrary domain scope expressions – meaning that designing and maintaining organizational schemas that map sets of users to resources is dependant on human administrators devising an organizational model and specifying it with meta-policies.

The reliance upon the flexibility of domain structures also creates several problems for administration. If objects can appear in multiple domains and policies apply in a cascading manner to domains and their sub-domains, the effects of updating policies, or of adding object references to domains, soon become difficult to predict. In order to analyze the effects of an update to a policy in the system, to check for policy conflicts and functional correctness, we need to identify all of the objects affected by the updated policy and all of the other policies that affect them. Any updated policy can effect every object in a domain and any of these objects may be referenced by an arbitrary number of other domains any of which can have policies applied to them. Furthermore, any policies which have subject or target domain-scope expressions which resolve to include any of the objects affected by the updated policy must also be considered. Finally, any meta-policies that relate to the updated policies, or to the sets of policies which

reference any of the affected objects must be considered. This presents us with a potentially large search space in order to identify the policies which must be evaluated in order to identify policy conflicts, and, depending on the sophistication of the algorithm that identifies policies that may be relevant and the rigor of the organizational schema defined by the administrators, we may have to perform many computationally expensive function calls in order to evaluate whether policies which contain arbitrary constraints conflict with the updated policy set for each object that is referenced by the updated policy. Although the rapid advances in the speed of modern computing hardware means that such computationally expensive processes can be unproblematic, policy based systems require high levels of responsivity and experiences of their deployment in industry show that policy sets tend to grow to large sizes quickly once they must deal with the quirks of real world organizations, making this a real handicap.

More problematic than the complexity of analyzing policy updates is the problem faced by the human administrator in mapping updates to the policy set into functional requirements of the system. Human administrators generally have a functional requirement in mind when they make updates to the policy set. In order for them to author policies to accurately reflect this function, they need to be able to understand the consequences of the update. The greater the complexity of the consequences, which can be expressed as the set of the policies that need to be considered, which is equal to the set of policies that apply to the set of objects referenced by the updated policy, the greater the difficulty in verifying that an updated policy will fulfill the administrator's functional requirement. In many cases, there is no way to automatically eliminate many policies from this functional verification as they relate to non-related constraints, which may still have a combinatorial effect which is undesirable from a functional point of view. Thus, while Ponder does provide us with abstractions which allow great flexibility in modeling an organization, in order to create a system that can more easily map a functional requirement to a policy specification, we need to devise an organizational schema which implements a more structured organizational model and limits the scope of the consequences of policy changes to a much more restricted set.

3. Community-Based Policy Management

The most basic element of our community-based model, which differentiates it fundamentally from the existing role-based approaches, is the fact that our model of the organization is primarily based on the organizational units that constitute it (e.g. divisions, departments, teams and working groups) and their relationship to the resources controlled by the organization. This contrasts with the focus on the positions of individuals within the organization and their relationships to resources used in the definition of roles. In our model, we view an organization as being composed of a hierarchical relationship of units, with the entire organization at the root of the tree and the most specific units within the organization as the leaves. Each of these organizational units has a goal with respect to its role as a unit within the organization. We view these units as the subjects of actions within the system and we analyze the emergent properties of the unit as a whole, independently of the properties of the individual actors that are members of the unit and might correspond to human users of the system.

We term these units *communities*, and though they may be typically formed around organizational functions, we would more commonly describe as structural units than functional units, according to the classification presented in [coutinho]. This high level approach to modeling organizations is most closely related to the minimalist approach to modeling human organizations – where “each level of the organization is treated as an organic entity, without regard for the specific details of methods and means by which its goals are attained” [cobb]. We choose the ‘community’ terminology firstly as it reflects the composite nature of units within human organizations, being made up of many human actors and secondly due to the fact that the term comes as close as possible to being a general purpose grouping abstraction which can encompass any type of human organization with a common high-level behavior, property or function. Or, to put it in plain English, communities are groups of individual people and they are generally groups with something in common.

An organizational unit is a composite, complex entity. Although we can speak of it as having a high-level function with respect to the organization to which it belongs, the successful performance of this function is dependant on the actions of the individuals who are members of it. However, as the function of the unit can not necessarily be derived from the functions of the individuals who make it up, the behavior or

properties of the unit are not a property of any single individual member, nor can they easily be predicted or deduced from the behavior of the members: they are emergent properties of the complex composite system that is the organizational unit. Models of organizations which are based upon the properties and behaviors of the individual actors within the system, such as role-based models, can not accommodate such aspects of organizations without introducing further abstractions and are then still faced with the problem of how to relate these abstractions to the underlying role-abstraction. Although TBAC and OrBAC both address this issue, they are both essentially extensions of RBAC, extensions which include an extra variable – team or organizational membership – in the assignment of roles and they do not attempt to model the emergent behavior or properties of the organizational unit. Our system is basically differentiated from such approaches due to the fact that we consider the organization primarily from the point of view of the composite organizational units that can be identified within it and model their properties and behaviors with respect to each other and the whole.

When we describe a community within an organization as the subject of actions within the system, we signify that we model it as an autonomous entity with emergent behavior in a number of ways. Most importantly:

- It has a relationship to the resources of the organization, autonomously of its membership. So, for example, the statement that the marketing department has a budget of \$100 million signifies a relationship between the functional unit denoted by marketing department and the financial resources of the organization. Since this relationship is not dependent on the properties of any of the individual members of the marketing department, we can say that it is autonomous of them and is an emergent property of the community as a whole. Modeling an organization in terms of communities and their relationships to resources allows this information – which is not derivable from the properties of any individual actor in the system – to be captured by our model.
- It is the subject of actions within the organization, autonomously of its membership. Thus, for example, we might say that the IT Department hired 20 employees. The action taken by the IT Department and the acquisition of the resources is not dependent on the properties of any of the members of the IT department and can be considered as an emergent action of the community.

In order for us to consider a community within an organization as being the subject of actions, we need to also consider it as being capable of taking autonomous decisions. Thus, we explicitly model the decision-making methods of each community. So, for example, we might say that gaining the approval of the Chief Technical Officer is the decision making method of the IT department, or that a vote of the board of governors is the decision making method of the entire organization.

A model of the organization in terms of communities possesses an inherent hierarchy, which we term a *hierarchy of authority*. This hierarchy is an effect of the compositional nature of organizations and the communities within them. On each level of the hierarchy, the communities are components of the communities above. This hierarchy imposes several conditions on the communities. Firstly, their membership, in terms of the individual actors who make up the community, is a subset of the membership of the communities above them. Secondly, the authority over resources possessed by a community is subordinate to the authority exercised in the community above it. Thus decisions made by the higher communities will over-rule those decisions made by the lower units, when they conflict.

A particular feature of our model is that we consider communities which exercise authority on behalf of other communities as communities in their own right and we position them in the hierarchy of authority beneath the community on whose behalf they exercise authority. Thus some of the communities which make decisions on behalf of larger units can be used. Modeling decision making units as simply another community in the hierarchy allows us to incorporate an organization's specific hierarchy of roles and positions into our hierarchy of communities without assuming any particular organization-wide model of decision making. So, for example, while some communities have dedicated subordinate units which handle decision making (executive groups), other may not and are therefore self-managed and may even be democratically run, using voting or consensus in order to reach decisions on behalf of the team.

We view a board of directors as a community which makes decisions on behalf of the entire organization. The decision making methods at the level of the entire organization are those that reflect the goals of the organization as a whole. So, for example, the board of directors of a publicly listed company take decisions on behalf of the entire company and these decisions are supposed to be aimed at maximize shareholder value. These decisions have an inherent precedence over decisions taken by communities further down the

organization's hierarchy. The IT department cannot veto the board's decision to outsource IT services. Similarly, the product development team cannot ignore the IT department's decision to cease product development. This hierarchy of authority springs naturally from the fact that the lower communities are components of the higher communities and their function is, by definition, to assist the higher communities in fulfilling their goals. As the goals of the organization changes, the behavior of the communities within it must change to reflect these goals.

Another aspect of this hierarchy that is important for the purposes of the organizational model relates to the relationship between the communities within the organization and the resources controlled by the organization (i.e. the resources to which the organization will apply policies). The root community in our hierarchy, which represents the entire organization, possesses authority over all the resources controlled by the organization. So, for example, the board of directors might decide, on behalf of the entire organization and in their capacity as the decision making mechanism for that community, that all accounts must be audited. The CTO does not have authority to over-ride this decision on behalf of the IT department with respect to that department's accounts. We say that authority for the accounting resources is rooted in the root community of the organizational hierarchy, or to put it another way, the community representing the entire organization possesses ultimate authority over the resource. In general, ultimate authority for all resources owned by the organization will be possessed by the root community, which represents the entire organization. This reflects the fact that the organization is the legal owner of the resources. Other communities within the organization possess limited autonomy with respect to these resources – their authority is bounded by decisions taken further up the community hierarchy.

Communities can possess authority to take decision on a subset of the resources controlled by the entire organization, an authority which they may be able to exercise autonomously of their parent in the tree of functional units. As we noted above, each community has authority over a subset of its parent's resources and this authority is bounded by the decisions of its parent. Thus, for example, the marketing department may have considerable latitude in deciding what to spend its budget allocation on, but be constrained by any policy decisions that the board of directors may have taken committing it to spending money in a particular way. For example, the board of directors could have taken a decision, on behalf of the entire organization, that only

approved suppliers and sub-contractors will be used. Any decisions taken by the marketing department will be constrained by this decision of the board. Similarly, the advertising team may have considerable decision making power when it comes to concepts, but be constrained to using printing contractors specified in a contract between the company and a partner.

It may be helpful in order to understand our concept of communities possessing authority over resources to consider the relationship of each of the communities in a simple organization model to a particular information resource, such as a database server. Since the server is the property of the entire organization, there are no limits on its authority over it. That is to say that, from an access control point of view, when we consider the entire organization as an agent, all operations are permitted. If the decision making mechanism for the root community, whatever it may be, decides that the community wishes to access the resource in a particular way it must be permitted by our access control system, otherwise the organization would not enjoy full control over its own resources. The other functional units of the organization may also possess authority over the resource, but this authority is constrained by any policies towards its use that may be defined by the root node of the organizational hierarchy. So, for example, while the IT department may possess authority to carry out a broad range of operations on the resource, creating and deleting databases without reference to the root node, they may be compelled to support database services for other departments and thus their authority over the resource is limited. For example, they may not possess authority to delete the data stored in a marketing database hosted on the server. Other departments may have even further limited authority over the resource, for example, the marketing department may have authority to read and update data in a certain database, but have no authority to define storage schemas or backup policies.

From the point of view of our model, we can also consider the hierarchical tree of communities to be a resource itself, also managed by the organization. This resource represents the structure of the organization and, similarly to any other managed resource, authority over this resource can be distributed throughout the organization. Thus, we can view organizational restructuring as simply another form of resource management. The community which represents the entire organization possesses authority to take any possible action on this resource – creating new communities beneath itself, or removing existing units for example. Furthermore, authority over a subset of this resource can be possessed by any of the

communities within the organization. So, for example, the IT department may possess authority to create new communities as subsets of itself – or, to put it another way, it may be autonomous from the rest of the organization with respect to its own internal organization. In practice, communities will often possess limited autonomy with respect to their own structure – with some latitude to establish new internal communities, while other aspects of its internal organization and structure will be dictated by the overall organization. For example, the IT department may be compelled by the organization's policy to have a community within it that is responsible for purchasing, while it may be free to decide on how it will divide itself into project teams.

Thus we state that a model of an organization based on communities as we have defined them contains an inherent hierarchy of authority. At the apex of this hierarchy is the community representing the entire organization, which has general authority over all decisions taken by all the individuals and units within the organization. As we descend the hierarchy, the communities have authority in more specific domains. We say that this hierarchy runs from the general to the specific. Rather than attempting to prove that this is a universally correct method of modeling organizations, we adopt it as our definition of an organization for the purpose of constructing an access control model.

To summarize, an organization is a hierarchy of communities, a hierarchy that runs from the general to the specific, where decisions taken by communities with respect to the organization's resources have a precedence that mirrors the hierarchy.

4. Case Study: RBAC Comparison

4.1 RBAC State of the Art

From the point of view of this research, which is interested primarily in the organizational model embodied by these access control models, it can be said that none of them provides an integrated view of the organization in functional or structural terms. For a start, none of the models attempt to fully integrate the model of how the system itself is administered into the model. This is either specified as a distinct administrative hierarchy, as in ARBAC97, or partially related to the user model through the organizational scope, as in AdOrBAC, or is left undefined and assumed to be carried out by the security officer. Furthermore, none of the models capture a great deal about the function of the system and how that relates to the functions of the constituent parts.

4.2 Case Study

This case study provides a comparison between the capacity of role hierarchies to capture organizational information and the capacity of CBPM to do so. The CBPMS is deployed in a simple software engineering company. This example is taken from the paper which formally defines the ARBAC model [sandhu99]. Note that although RBAC is often considered not to belong to PBM, due to the fact that the standard RBAC model only supports simple positive permissions, from the point of view of this work, permissions can be considered to be policies written in a very simple policy specification language (which allows a single policy to be specified - permit). Thus RBAC systems can be considered to be a subset of PBM. A number of roles are identified within the organization, namely *Director*, *Employee*, *Engineer*, *Production Engineer*, *Quality Engineer* and *Project Lead*. In addition to these basic roles, several project-based roles are required, since the permissions of each individual are dependant on which project they have been assigned to. For simplicity, only two projects are modeled as this illustrates the principle adequately. Finally, a set of administrative roles are also required for the management of the system, namely *Senior Security Officer*, *Department Security Officer* and a *Project Security Officer* for each project. Using these roles, the role hierarchy pictured in figure 1 is constructed. From the point of view of the CBPMS, integrating a permission language is trivial.

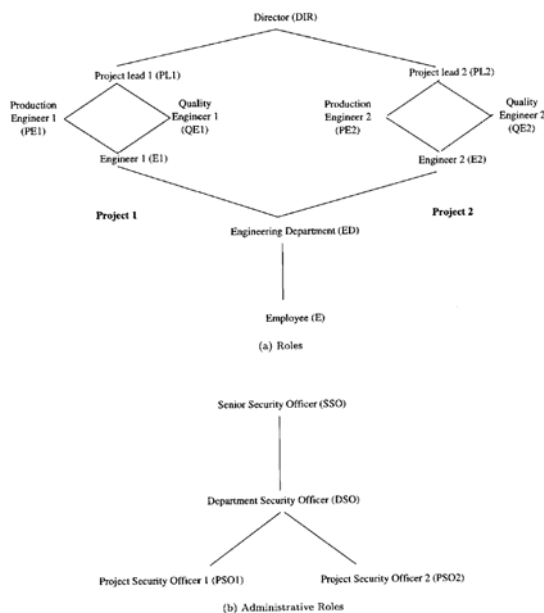


Figure 1 RBAC / ARBAC role hierarchy

This role hierarchy is intended to model a situation where there are two separate projects within an

organization which require different access control policies for those working on them, and there are a variety of different roles within each project. There are many problems associated with the use of role hierarchies to model organizations in policy based systems. The multiplicity of distinct role hierarchies within organizations identified by [moffet98] means that in order for role hierarchies to be applied to real-world organizations while respecting important security principles such as the principle of least privilege, the hierarchical structure must be adapted to include complex constructs such as private roles and hierarchies and limited inheritance [sandhu96]. Thus, for example, in the above hierarchy, it is generally not desirable for the *Director* to inherit all the permissions of every engineer in the organization, and thus private versions of the various engineering roles will have to be created. The need for these complex role-models stems from the fact that the hierarchical relationships in the model express different relationships. While a *production engineer* and a *quality engineer* are genuinely specializations of the engineer role – in accordance with the notion of class hierarchies in object oriented programming – the *Director* and *Project Lead* roles are not specializations of the roles that they inherit. Their relationship to the roles beneath them is a relationship of authority, not of specialization. The roles beneath the director must follow her decisions and that is what these relations represent.

Furthermore, from a glance at the role hierarchy, it is immediately apparent that the organizational model is divided in two – administrators of the system and users are modeled in separate hierarchies. This combines with the earlier problem to create an organizational model which is at once overly complex and also fundamentally fragmented.

By contrast, the community based model is designed in order to incorporate a naturalistic model of the organization. Rather than simply representing atomized roles, communities also contain a collective identity. This collective identity is expressed in the authority possessed by the community (through delegations) and can also be expressed through policies which contain collective decision making conditions. Furthermore, since the community structure itself is simply another resource, there is no need for a separate administrative hierarchy. From this basis, the role hierarchy above can be remodeled in order to take advantage of these features without sacrificing its ability to represent the organization.

Figure 2 shows the remodeled community hierarchy. In general, this follows the CBPMS approach to modeling organizations, where a

community is positioned as an immediate sub-community of that community which is its smallest super-set in terms of membership and authority. In this case, the *Employees* community is the smallest super-set of the *Director* community in the hierarchy, so it belongs as an immediate sub-community of *Employees*. In general, this rule holds true for the community hierarchy – communities representing positions of authority over other communities are positioned as immediate sub-communities of that community. This introduces an important divergence from the hierarchical RBAC paradigm. In the RBAC example cited, the *director* role inherits all permissions or policies specified for the other non-administrative roles in the organization. Thus, if a policy is added to the *Project 1 Engineers* role, this policy will automatically apply to the *Director* role too. In the modified community model, this is no longer the case. If a policy is added to the *Project 1 Engineers* community this will not automatically apply to the member(s) of the *Director* community since it is not a sub-community of *Project 1 Engineers*. In order to make the new policy apply to the members of the *Director* community, the policy must also be added to that community, or must be specified in their closest common ancestor – in this case the *Employees* community.

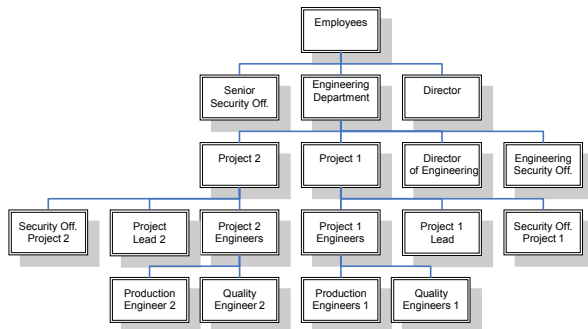


Figure 2 Community Hierarchy

Meanwhile, the administrative hierarchy has been integrated into the organizational model. Thus, for example, the *security officer* of each project is a sub-community of the project itself and the *senior security officer* is a sub-community of the *employees* community. In order to use this community hierarchy to capture the requirements of the organization, and in particular to model the administration of the community structure itself, resource authorities representing the community structure resource are distributed throughout the organization in the following way.

- The community resource representing the employees, engineering department, and the project communities are delegated to the communities themselves – giving each of these communities limited self management. Then this authority is further delegated to the relevant administrators, who will carry out the self-management on behalf of the community above (i.e. they are control communities).
- A policy is created for each one of these control communities which specifies that the administrator is permitted to carry out any changes that are approved by the community that manages the community above – thus, for example, the senior security officer requires the approval of the director for changes to the policies which affect all employees, while the approval of the *project 1 lead* is required for changes to the policies which affect project 1.

This creates a situation where decision making authority is distributed so that the following rules apply:

- The *Senior Security Officer* can define policies, with the approval of the director, which will apply to all employees in the organization. Due to the fact that policies are evaluated according to the hierarchy of authority of the organization, policies defined at this level can not be over-written by policies defined further down the organization.
- The Engineering Department’s security officer can define policies, with the approval of the director of engineering, which will apply to all members of that department – as long as they do not conflict with policies which apply to all employees.
- The security officers for the two projects can define policies, with the approval of the project leads, which will only apply to members of their respective projects and these policies will be bound by any policies defined for all employees, or for all

This hierarchical distribution of decision making authority captures the administrative requirements of the organization in a much more structured way than the administrative hierarchy in the role-based version. The community structure tightly couples each organizational unit with its decision making process and the limited delegation of authority over the community resource ensures that administrators are limited to specifying policies which apply to the organizational units for which they are responsible.

In addition to the communities which have been more or less directly mapped from the role hierarchy

into this modified community structure, several new communities have been introduced. In particular a community represents each project. These are grouping communities which may not have any policies specified within them and may merely serve as a conduit for distributing authority throughout the organization. The individual permissions of users are defined by policies set in sub-communities, while the grouping communities merely serve as delegation distribution points for resources, gathering together all the authority that is required by all the communities underneath them. Hierarchical roles are not suitable such grouping and authority distribution constructs. Their support in community models facilitates a much more organic model of the organization, by grouping related communities together, even in cases where they share no policies, as well as facilitating the principle of minimum delegation of authority.

The emulation of flat RBAC above required that authority for all managed resources be delegated from parent to child throughout the community hierarchy. This removed any limitations on the policy scope of each community, allowing policies on all targets and for all actions managed by the system to be specified for each community. The only exception being that authority for the community resource itself was granted to the organizational administration community. Otherwise each community had the same authority as its parent community and delegates all of this authority to its sub-communities.

One advantage that the community model provides is, thus, the ability to delegate a limited sub-set of authority to a sub-community, thereby limiting the target-scope of policies defined within that sub-community. So, for example, rather than delegating all authority for all resources from the *Engineering Dept.* community to the *Project 2 Engineer* community, the community model operates on the basis that only that authority which is required to fulfill a community's function is delegated. For example, if the only managed resource used by Project 2 is a particular database, then authority for the particular database is delegated rather than authority for all resources. This principle of minimum delegation provides several advantages over the non-scoped RBAC approach. Firstly, the danger of erroneous policy specification granting excessive access rights to individuals is reduced since any policy within a community which has a target that has not been delegated to the community will not be accepted by the system. In a role based system, there is little protection against a stray policy specification which grants un-envisaged access to the holders of a role which should only grant minor privileges. The CBPMS guards against this by

requiring an explicit delegation before any policies can be authored with the resource as their target.

Secondly, the distribution of authority over resources – which resources groups are allowed to use and in what ways they can use them – is mapped across the organization. By following the chain of delegations throughout the community model, an informative overview of resource usage and requirements is contained within the model.

Finally, the CBPMS policy search algorithm utilizes the delegations of authorities to eliminate branches of the community hierarchy from the policy search. When a policy decision request arrives at the service, referencing a particular target resource, the CBPMS only needs to search for applicable policies within communities which have been delegated the required authority over the relevant resource. In a role based system, each role that the subject is a member of must be searched for relevant policies since there is no way of knowing if a role might contain an applicable policy for a particular decision request without examining the policies within it. Furthermore, the CBPMS conflict resolution approach allows all sub-communities to be discarded from the search as soon as an applicable policy is discovered in a parent community. In complex organizational models, this can vastly reduce the size of the policy search space.

An important feature supported by the community model which is absent from the role-based model is the concept of self-management. Since the community structure is modeled as a resource like any other, authority for community management can be distributed throughout the organization like any other resource. Thus, a community may have authority to author its own policy rules and to spawn sub-communities and delegate authority to them. However, any policy rules or delegations which target resources which have not already been delegated to that community from its parent will not be accepted by the system. This effectively allows the organization to be modeled in such a way as to support bounded self-management. The delegations define the boundaries, and within those boundaries the community is self-managed. By contrast, role based models offer none of these features.

Although the community model's support for automatic conflict detection based on the community hierarchy has been touched upon above, it is worth making a few final comments on this feature. For a start, it might seem somewhat counter-intuitive that the conflict resolution algorithm always follows the community hierarchy. This means that policy rules specified in the *Employees* community will have precedence over those specified in the *Director*

community which inverts the normal concept of an organization's hierarchy. However, it should be recalled that when a policy is specified in a community, this signifies that the policy should apply to all members of the community without exception. Thus, care must be taken so that any policies specified for a community should really apply to all members of the community. In many cases where a policy should only apply to a subset of a community, but there is no community to represent this subset, this signifies an incomplete community model and an intermediate community is required in order to incorporate the required subjects.

In addition to the simple parent-child automatic conflict resolution mechanism, the CBPMS can also detect and resolve conflicts between communities which have no direct hierarchical relationship by ascending the community tree to the nearest common parent of the conflicting communities and applying that a child combination algorithm. This detailed conflict detection and resolution mechanism can also be employed at policy-specification time in order to highlight situations in which specified policies will be overruled by others defined elsewhere in the community hierarchy.

6. Conclusions and Further Work

In this work we demonstrate that community based policy management offer more flexibility in the management of resources and organizational structure in dynamically changing organizations than existing role based abstractions. In our future work we aim to empirically verify the benefits of community based access control through emulation of policy authoring patterns.

9. Acknowledgements

This material is based upon work partially supported by the Higher Education Authority under the M-Zones programme.

10. References

- [bertino] Bertino, E., Bonatti, P. A., & Ferrari, E. (2001). TRBAC: A Temporal Role-based Access Control Model, *ACM Transactions on Information and System Security*, 4(3), 191-233
- [boros] Boros, S. Policy-based network management with SNMP, Scientific Output of the University of Twente, The Netherlands, 2000.
- [cohen] Cohen, E., Thomas, R. K., Winsborough, W., and Shands, D. 2002. Models for coalition-based access control (CBAC). In Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies (Monterey, California, USA, June 03 - 04, 2002). SACMAT '02. ACM Press, New York, NY, 97-106. DOI=<http://doi.acm.org/10.1145/507711.507727>
- [coutinho] Coutinho, L. R., Sichman J.S., Boissier O. Modeling Organization in MAS: A Comparison of Models First Workshop on Software Engineering for Agent-oriented Systems, SEAS 2005
- [cuppens] Cuppens, F., Mieke, A. AdOrBAC: an administration model for Or-BAC *International Journal of Computer Systems Science & Engineering*. Vol. 19, no. 3, pp. 151-162. May 2004
- [damianou] Damianou, N., N. Dulay, E. Lupu and M. Sloman (2001). The Ponder Policy Specification Language. In Proceedings of the Policy Workshop 2001, HP Labs, Bristol, UK, Springer-Verlag, 29-31 January 2001.
- [doeppner] Doeppner, W. T., Glacalone, A formal description of the UNIX operating system. Proceedings of the second annual ACM symposium on Principles of distributed computing, 1983.
- [el-kalem] El-Kalam, S. Benferhat, A. Mieke, R. El-Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization based access control. In POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, page 120, Washington, DC, USA, 2003.
- [feeney04] K. Feeney, D. Lewis, V. Wade, Policy Based Management for Internet Communities, in Proc of IEEE 5th International Workshop on Policies for Distributed Systems and Networks (Policy 2004), June 8th-9th 2004, IBM Thomas J Watson Research Center, New York, USA, pp 23-34
- [feeney05] K. Feeney, K. Quinn, D. Lewis, D. O'Sullivan, V. Wade, Relationship-Driven Policy Engineering for Autonomic Organisations, in Proc of IEEE 6th International Workshop on Policies for Distributed Systems and Networks (Policy 2005), June 6 - 8, 2005, Stockholm, Sweden
- [feeney07] K. Feeney, D. Lewis, V. Wade, Roles Considered Harmful in Policy-based Management for Dynamic Organisations, to appear in proc of Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)
- [georgiadis] C.K. Georgiadis, I. Mavridis, G. Pangalos, and R.K. Thomas, "Flexible Team-based Access Control Using Contexts", Proceedings of the 6 ACM Symposium on Access control models and technologies, May 2001, pages 21-27
- [lewis06] D. Lewis, K. Feeney, K. Foley, L. Doyle, T. Forde, P. Argyroudis, J. Keeney, D. O'Sullivan Managing Policies for Dynamic Spectrum Access, in Proc of IFIP TC6 1st Autonomic Networking conference, Paris France, September 2006
- [lewis04a] D. Lewis, K. Feeney, K. Carey, T. Tiropanis, S. Courtenage, Semantic-based Policy Engineering for Autonomic Systems, in Proc of 1st IFIP WG6.6 International Workshop on Autonomic Communication - Autonomic Communication Principles, Berlin 18-19 October 2004.
- [lewis06] D. Lewis, K. Feeney, K. Foley, L. Doyle, T. Forde, P. Argyroudis, J. Keeney, D. O'Sullivan, Managing Policies for Dynamic Spectrum Access, in Proc of IFIP TC6 1st

Autonomic Networking conference, Paris France, September 2006

[lupu] Lupu, E. C.. A Role-Based Framework for Distributed Systems Management. PhD Thesis, Department of Computing, Imperial College. London, U. K., July 1998

[sandhu00] Sandhu, R., D. Ferraiolo and R. Kuhn. The NIST Model for Role-Based Access Control: Towards A Unified Standard. In Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, pp. 47-61, 26-28 July 2000

[lupu] Lupu, E. C.. A Role-Based Framework for Distributed Systems Management. PhD Thesis, Department of Computing, Imperial College. London, U. K., July 1998.

[moffett] Moffett, J. D., Control Principles and Role Hierarchies. Proceedings of the Third ACM/NIST Role Based Access Control Workshop, Fairfax, Virginia, USA, ACM Press, October 1998. pp 22-23

[sandhu96] Sandhu R., Coyne E., Feinstein H., Youman, C. "Role-Based Access Control Models," Computer, vol. 29, no. 2, pp. 38-47, Feb., 1996.

[sandhu99] R. Sandhu, V. Bhamidipati, Q. Munawer. The ARBAC97 model for rolebased administration of roles. ACM Transactions on Information and System Security, 2(1):105--135, 1999.

[sandhu00] Sandhu, R., D. Ferraiolo and R. Kuhn. The NIST Model for Role-Based Access Control: Towards A Unified Standard. In Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, pp. 47-61, 26-28 July 2000

[sloman94a] Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Plenum Press. Vol.2 No. 4, 1994

[sloman94b] Sloman, M. and K. Twidle. Domains: A Framework for Structuring Management Policy. Chapter 16 of Network and Distributed Systems Management (Sloman, 1994ed), pp. 433-453

[taylor] Taylor, J. G. (1962). The behavioral basis of perception. New Haven: Yale University Press.

[Thomas] Thomas R., TMAC: A primitive for Applying RBAC in collaborative environment," in 2nd ACM, Workshop on RBAC, FairFax, Virginia, USA, November 6-7 1997, pp. 13-19.

[thompson] Thompson P. and McHugh, D., Work Organisations, Palgrave, New York, 2002.