

Toward Explicit Policy Management for Virtual Organizations

Glenn Wasson
wasson@virginia.edu
Department of Computer Science
University of Virginia
Charlottesville, VA 22904

Marty Humphrey
humphrey@cs.virginia.edu
Department of Computer Science
University of Virginia
Charlottesville, VA 22904

Abstract

A Virtual Organization (VO) is a dynamic collection of distributed resources that are shared by a dynamic collection of users from one or more physical organizations. As Grid Computing technology is starting to facilitate truly large-scale VOs, issues are being raised regarding the purpose, architecture and operational mechanism of the VO. The emerging approach is essentially to define the VO as a particular set of users, whereby a "VO server" issues tokens to humans attesting to their membership in the VO. The problem with this approach is that there is little in the way of rules that describe the operation of the virtual organization or rules that govern the behavior of VO users and resources (and the ramifications of failing to meet the intent of the VO itself). Where such rules exist, they are implicit and therefore difficult to enforce in a consistent or automated manner. This paper identifies two representative policies for existing and future VOs and, more generally, identifies issues and approaches for addressing the practical concerns for implementing any explicit VO policy: utilization measurement, accounting, enforcement conditions, enforcement actions, and security. A prototype implementation using .NET is described.

1. Introduction

A Virtual Organization (VO) is a dynamic collection of distributed resources that are shared by a dynamic collection of users from one or more physical organizations. In this paper, we focus on a common type of virtual organization that is formed to tackle large-scale scientific problems, e.g. [1][3][11][13][15]. Large computing centers typically provide the resources, and domain scientists are selected as users. The emerging approach in Grid Computing [10] is essentially to define the VO as a particular set of users, whereby the equivalent of a "VO server" issues tokens to humans

attesting to their membership in the VO [5][21]. These tokens are then presented to the individual resources.

While some VOs provide resources to any number of projects (i.e., resources are not devoted 100% to the VO), others like ATLAS Grid [3] are devoted to a single project. The ATLAS VO is divided into sub-organizations (tiers), each of which is devoted to processing the output of the previous tier and providing the results as input to the next tier. Such an organizational structure is created because the computational load at each tier must be distributed across the tier sites in order to perform the work in a timely manner. In essence, the virtual organization's policy is that each site in the VO should perform $1/N^{\text{th}}$ of its tier's workload.

However, current VOs rely on the individual sites to take on and complete their share of the overall VO workload. But what if some sites do not conform to the policy, either through oversight or malice? This results in both increased, unexpected load on the other resources in the VO and an increase in the time needed to complete the work. The entire virtual organization is effected as some sites have less of their resources to devote to non-VO projects while other sites sit idle waiting for data. The scale of current scientific collaborations may allow for all sites to be implicitly trusted to implement a virtual organization's policy, but as VOs grow in size and scope, this model will become less effective. As VO member institutions and users become increasingly diverse, conflicts between (1) the goals of the VO, (2) the goals of its member resource providers and (3) the goals of its users will become increasingly common and therefore increasingly difficult for VO administrators to solve at the bureaucratic level or even recognize at all.

To handle virtual organizations of ever increasing scale, explicit policy is required for governing VO operations. Having (explicit) policy for each member of three identified classes (VO, resource providers, users) not only allows VO members (i.e., resource providers and humans) to know what will be expected of them before

joining, but it provides a structure around which to base policy enforcement. Since enforcement is of crucial importance to the value of policy-based VOs, we consider its implications on VO architecture as well as VO members. The contributions of this paper include a discussion of general issues in virtual organization policy, the definition of two specific VO policies and their implications, and a prototype policy-based VO implemented in .NET. The remainder of this paper is organized as follows. Section 2 discusses the types of policies that virtual organizations may require. Section 3 discusses two specific VO policies that may be of interest to current virtual organizations and the implications of these policies on system architecture. Finally, section 4 presents a .NET-based grid architecture that facilitates such policy governed virtual organizations.

2. Policy in Virtual Organizations

Policy for virtual organizations can cover almost every aspect of the VO including statements of its purpose, its day-to-day operation and what is expected of various members (both resources and users). Here, we divide VO policy into three categories.

- VO-wide operational policy
- VO policy on resources
- VO policy on users

VO-wide operational policy consists of statements about the intended operational state of the virtual organization. VO-wide policy statements are intended to describe the state of the VO as a whole and not any single site or service. In other words, this type of policy describes the distribution of resource utilization across the whole VO. For example, a policy that states “the compute load of the virtual organization is to be divided equally among all member sites” describes the VO’s intended steady-state. Other policies might include:

- All work is to be performed on large queuing systems from 9 am – 5 pm and on PC clusters after hours.
- 75% of the VO’s data will be stored in the VO archive. The remaining 25% will be evenly distributed through the member storage resources.

While making these types of broad statements explicit may benefit resources in their decision about whether to join the VO, we expect policy statements to be much more concrete—in effect, dictating policy by describing actions that will be taken to maintain the desired VO-state. For example, a policy statement such as “if any site is performing less than 25% of the work of the other sites, all new work will be scheduled on that site until the work load is equalized” describes an explicit trigger condition

and an action that will be taken if that condition is met. Note that while such policies may allow for automated responses from the VO to specified conditions, these responses need not be “fully automated”. That is, the system’s response could be to contact the appropriate administrator and alert them to the condition or to suggest a set of corrective actions to a human who edits and chooses when to enact them.

Note that VO-wide operational policy is different than policies used in [12], [16], and [18]. These systems make permit/deny decisions based on (potentially multiple) access control policies and the accessor’s identity / group membership. VO-wide operational policy refers to how a VO will allocate its resources given its workload. Also, while it is possible to implement some VO-wide policies by having each resource enforce the same policy, other VO-wide policies may call for the resources to implement different policies to achieve the desired (VO-wide) effect. For example, the policy that non-secure transactions are processed by resources A, B or C and secure transactions by resources D, E, and F divides the VO into resource groups which locally enforce different policies.

VO resource policy describes the virtual organization’s rules for the behavior of its member resources. This type of policy is useful for setting rules that pertain to particular resources within the VO rather than across the entire organization. For example, “the chemistry department must provide the university VO with dedicated compute time from midnight to 6 am” or “the astronomy department must provide the VO 20% of its telescope time”. Again, policy should be stated in terms of rules that describe the process of policy enforcement. When crafting such rules, it is important to understand the motivation of a resource in joining the VO. Why does the astronomy department want to provide telescope time to people outside the department? The answer may be because of other agreements between the VO and other resource providers, e.g. the computer science department must provide 500 GB of storage to the VO. Depending on the VO-wide policy, this storage may be available for the astronomy department as a VO member. If one resource is not fulfilling the VO’s policy, that site can receive correspondingly less access to the remaining resources in the VO. Today, the motivation for resource providers to make their resources available to scientific VOs is often an informal agreement indirectly related to the advancement of some funded project. While the coarse-grained nature of the payment in this model presents many enforcement challenges, future virtual organizations may provide payment to sites on a much more fine-grained basis. This will provide more potential points of enforcement for the system.

VO user policy is the third type of policy, which specifies rules for users of the virtual organization's resources, again, from the perspective of the VO itself (not from the perspective of the user). These rules need not be permit/deny rules on various possible VO-wide user actions [11], but may specify pre-conditions for any given user action, as well as obligations that the user receives for performing an action. Preconditions and obligation in policy [4][7] provide a compact means of representing what a user must do *before* performing a specific action as well as what they must do *after* an action. The exact definition of "action" will be specific to a given VO. It may be simply invoking a remote method on a grid service [8], or it may be performing a specific role in a long running, transaction process.

There is a connection between the various types of VO policy in that VO-wide policy can be specified by applying the same resource and/or user policy to all resources/users in the VO. In other words, if each site in a 4 site VO has a resource policy that it must perform 25% of the VO's total workload, this is equivalent to a VO-wide policy that all sites will equally divide the VO's work.

Note that the work described in this paper is focused on the VO level. Resource providers will undoubtedly have their own policies governing their actions, e.g. "provide 50% of a machine's cycles to grid user jobs and 50% to local user jobs". Users may even have their own policy, specifying their preferences for actions that the system will ultimately take on their behalf, e.g. "I prefer resource 1 when available, but otherwise want to run on the fastest available machine". Managing conflicts between these policies and VO policies is a rich area for future research and is not addressed here.

3. Representative VO policies

In the remainder of this paper we focus on two VO-wide policies that we believe are implicit in today's virtual organizations. Our purpose is not to outline a specific policy language, but rather to discuss the implication of certain policy decisions on system architecture and implementation. For the policies presented below, we consider questions of utilization measurement, accounting, enforcement conditions, enforcement actions and security. In Section 4, we address these concerns in our prototype policy-based grid system implemented in .NET.

Policy 1: All resource utilization is divided equally among the member resources [the 1/N policy]

We refer to this policy as the "*1/N policy*" because each resource in the VO is to perform $1/N^{\text{th}}$ of the work of the VO. This policy can apply to any resource which is distributed throughout the VO's member organizations, cycles, disk space, or other specialized resources. A 1/N policy is commonly used in scientific VOs, but is neither explicitly stated in policy nor enforced.

Policy 2: Each user receives VO utilization credit equivalent to the resource utilization they provide to other users [the you-get-what-you-give (ygwyg) policy]

Instead of requiring an equal distribution of resource utilization throughout the VO, this policy allows for users to utilize as much of the VO's resources as they wish provided they "repay" the VO by providing access for other members to resources they control.

3.1. Utilization measurement

How can a policy enforcement service determine if a VO's policy is being fulfilled by the VO's member resources? To measure utilization for a *1/N* policy, a service must be able to assess the total demands on the VO for a particular resource and the current utilization at each member resource. To measure utilization for a *ygwyg* policy, a service must determine the resources being consumed by a particular user and the resources being provided to other VO members by that user.

For some resources requests (e.g., storing a file) the most effective way to determine the total demands throughout a VO is to have a single VO-wide interface through which users access that resource. For example, this is analogous to a Web-based portal by which users "upload files to the VO" from their desktops, and the portal itself determines where in the VO to actually place the data. This provides a single point through which all resource requests flow and so total resource utilization is the sum of all granted requests. For other resources requests (e.g., running a job) a centralized interface is less effective because of the difficulty in knowing how much of the resource will be consumed by the request a priori. Each job's final utilization will still need to be read from the specific resource assigned to process it. In still other situations, existing daemon-based approaches are the best available approach to measure resource utilization. That is, a root process constantly monitors the resource and records the resource consumption. This is particularly attractive for legacy applications that must "run in the VO" but cannot be (or should not be) modified to accommodate measurement. However, VO resource consumption can be more complicated than a simple sum

over all resources. For example, it might be necessary to measure the *coordinated usage* over multiple resources, if the VO's policy is to "reward" such efforts (under the assumption that the VO's mission is being accomplished under such conditions).

While transparent access to VO resources (i.e. the user is not necessarily aware of the physical resource being used for their request) is a goal of grid systems, very few existing VOs have a centralized portal through which all resources are accessed. This work does not advocate the creation of a centralized system. In fact, many of the policy issues raised here result from the fact that there is no single best place to encode a virtual organization's policy. One solution that does not involve a centralized resource access point is to require that any resource joining the VO run a specialized service which provides utilization information to the VO's accounting service.

Actual resource utilization may not be the only parameter that must be measured. For example, in a VO with a *ygwyg* policy, does a user providing access to their PC-cluster actually need other users to run on the cluster in order to receive credit or can they receive credit merely by making the resource available? Clearly time of day and length of contiguous availability are important also. Perhaps a VO also cares about reliability or security of the resources. In general, many parameters could be measured. It is useful to think of resources as providing a certain quality of service (QoS) to the VO's users, with the VO policy dictating how this is to be measured.

3.2. Accounting

For any VO policy, some form of accounting service is needed to make use of the utilization data. The collection of distributed resource QoS information for the accounting service can be handled by services such as the Globus project's MDS [7]. The accounting service compiles the utilization information so that the enforcement system can check for enforcement conditions.

One important issue in accounting is the "exchange rate" for various resources. How can the utilizations of CPU time on various processors be compared? How much disk space does a user in a *ygwyg* VO need to donate to get 3 CPU hours on another machine? While VO policy needs to specify this, it is beyond the scope of this work. It is however being addressed by other projects in the Global Grid Forum [19].

3.3. Enforcement Conditions

Enforcement conditions are the conditions under which the VO should take some action to enforce its policy. VO policy should not only specify nominal distributions for utilization of various resources (e.g. $1/N$ or *ygwyg*), but also some tolerance that resources must stay within to comply with the policy. These tolerances represent a set of enforcement conditions that will trigger enforcement actions.

A larger issue is deciding who gets credit/punishment for complying/not complying with VO policy. For a $1/N$ VO, resources are thought of as purely providing services to the VO but not demanding any services of it. This makes it difficult to enforce policy on a resource because there is no suitable punishment for non-compliance (but see the discussion of correction vs. punishment below). However, VO users are thought to only consume resources and not provide any. Ultimately, it must be the VO users who are punished for a resource's non-compliance. The question then becomes which users to associate with a given resource? If a super-computing site contributes its resources and its users to a VO, those users can be credited/punished if the site does not meet the VO policy (even if a particular individual had nothing to do with non-compliance). In a *ygwyg* VO, this question still exists. Even though the distribution of resource utilization is not equal as in a $1/N$ VO, there is a particular user (or set of users) who receives "credit" for the QoS provided by their associated resource.

VO Policy also needs to specify the effect of an enforcement condition being detected on in-progress operations. Should an ongoing user operation be immediately terminated or allowed to complete when an appropriate enforcement condition is detected? What should be done with the (possibly partial) results of that operation? Should they be returned to the user, removed or held in escrow until the enforcement condition no longer exists?

Another matter is how conditions in the VO at startup affect enforcement conditions. In a $1/N$ VO, policy should specify some minimum VO-wide workload below which the policy is not enforced. This prevents enforcement actions from being taken when there is not yet enough demand to warrant a change in the utilization distribution. In a *ygwyg* VO, the policy must specify the initial credit that is given to a new user. Such a VO depends on each user both consuming resources and providing them. If a user only provides resources that other VO members use, but does not consume any, eventually that user will receive a disproportionate

amount of the total “credit” in the system. This can have the effect of starving out other users. Initial credit must be carefully selected to not allow users to perform large amounts of work without having to contribute resources, but not so small as to allow the VO to easily bog down when one user does not currently require resources.

It should be noted that *ygywg* VOs have an inherent “risk”. The VO’s enforcement conditions must define a maximum possible “debt” that a VO member can accumulate before being completely denied access to VO resources. The virtual organization “risks” this much resource utilization whenever a new member joins the VO because that user could potentially consume that much of the VO’s resources. If the user then leaves the VO (with their associated resource), the VO has lost the ability to collect service back from that user (see section 4.3 for a discussion of another similar kind of risk). Obviously decisions on VO membership must be made carefully as members who can derive long-term benefit from access to a VO’s resources are more likely to stay in the organization.

3.4. Enforcement Actions

The two basic questions for a VO’s enforcement actions are precisely what those actions are and who performs them. There are two types of actions that can be taken, punitive and corrective. Punitive actions are those which reduce the quality of service that the VO provides to a user or set of users. The idea is to cause those users to enact a change in a particular resource’s compliance. Corrective actions are actions in which the system attempts to alter the VO’s distribution of resource utilization to achieve compliance. For a 1/N policy, a corrective action might be to migrate files if one site is storing less of the VO’s data than other sites. Note that corrective actions are not meant to “punish” under utilized resources, but to alleviate the load on over utilized ones.

In general, whether a policy specifies punitive or corrective actions will depend on the ability of the VO to affect the individual member resources. The more independent the member resources are, the less likely it is that an enforcement service can effectively correct for policy non-compliance. For example, if VO members can directly submit jobs to VO resources (as opposed to using a centralized scheduler), then those compute resources must be willing to accept a directive from an enforcement service to re-route submitted jobs to an under utilized resource. Otherwise, policy enforcement can be circumvented. Corrective actions will also be limited by

the requirements of any particular resource request. For example, not all compute nodes in a VO will have the same architecture and so it is not possible to route job requests to arbitrary nodes, even if they are under utilized. This brings up an issue that is particularly germane to *ygywg* VOs. The VO policy should specify some minimum set of requirements for any user/resource pair to join the virtual organization. If a resource is sub-standard compared to others in the VO, no user will want to use that resource, and so its owner, will be unable to receive credit for its use.

VO policy will likely have multiple levels of enforcement actions depending on how far a particular resource is from compliance. Corrective actions in a 1/N VO will depend on the resource that must be equally distributed. To more evenly distribute compute cycle use, new scheduling approaches can be used to direct jobs to resources that must perform more work. To move evenly distribute space utilization, files can be migrated between storage resources after they are initially placed. In *ygywg* VOs, corrective actions will typically involve some form of scheduling as various resources will need to attract use for their owners to receive credit. Some resources may require assistance from a VO level scheduler to prevent starvation. The most obvious punitive action is to restrict a particular user’s (or set of users’) access to other resources in the VO if their associated resource is non-compliant. Other possible punitive actions include sending email to the appropriate user or administrator or allowing users to make requests of the VO’s resources, but holding the output data in escrow until the appropriate resource reaches compliance.

The final question is who actually performs the enforcement actions? While an explicit VO policy allows for automated enforcement, this need not be the case. In many cases, a human may be in the loop to authorize/schedule an enforcement action. No matter how humans are involved, there may be multiple enforcement services. Policy may specify that there is one per resource, one per user, one per action, or some other measure. The selection of the number of independent enforcement services will depend on which method will most effectively scale with the virtual organization.

3.5. Security

An important issue for both of the VO policies presented here is how any of the VO system services can ensure the veracity of messages they receive. As VOs grow larger and contain more members, there is a greater chance for accounting services to receive bogus utilization information or for enforcement services to receive bogus

directives about a user or resource's non-compliance with VO policy.

One solution is to use digital signatures on messages exchanged between services. The WS-Security specification [2] defines a standard technique for signing XML messages. PKI cryptography and X.509 certificates can be used to generate signatures and different portions of the same message can even be signed by different entities. For example, any resource utilization report sent to the accounting system by a resource should be signed by that resource's private key. Note that this implies that each resource has its own certificate. While this signature would prevent a third party from modifying the utilization report on the wire, the resource itself may receive bogus user requests. This requires that all user requests be signed by the user, enabling the resource to verify that they are a member of the VO. Timestamps can be used to prevent third parties from artificially inflating a resource's utilization by replaying valid user request messages.

There is a similar problem with messages sent to the VO's enforcement service. Any data that this service receives from the accounting service must be signed by the accounting service. Any resources that an enforcement service contacts to restrict a member's privilege must similarly expect those messages to be signed by the enforcement service.

4. Prototype System

We have implemented a policy-based VO prototype using Microsoft's .NET and Web Service Extensions (WSE). This grid-system is composed of a set of web services that act as access points to resources, handle resource utilization accounting, and enforce the virtual organization's policy. The prototype system described here implements the *ygwyg* policy.

The policy-based VO prototype consists of 3 types of web services: *GateKeepers* that represent access points for resources in the VO, *Enforcers* that are in charge of carrying out the VO's enforcement actions, and a *Bank* that collects resource utilization data and holds information about member users and resources. All of these services are stateful¹ and expect any request

¹ Note that although web services are traditionally thought of as stateless, the .NET framework web services can actually maintain state between invocations and expose that state to other services. This is accomplished through

messages sent to them to be digitally signed by an authorized entity.

4.1. The GateKeeper service

The GateKeeper service is similar in spirit to the Globus Gatekeeper and provides access to a resource for VO members. In the prototype, members can request use of processor and disk resources from the GateKeeper service. The GateKeeper does not place any limitations on the use of a resource once an operation (running a job or writing a file) has begun though this will be addressed in future versions. The GateKeeper determines resource utilization after the operation is completed and sends that information to the Bank. Since the prototype VO implements the *ygwyg* policy, a GateKeeper also has an associated user who is credited for service performed by the GateKeeper. Currently, a resource is associated with exactly one human. However, this is sufficient for this prototype because the resources are desktop PCs.

The GateKeeper interface provides three important methods: `Register`, `RunJob` and `WriteFile` (additional methods to read and remove files including job results are not discussed here). The `Register` method is used when joining the VO. When a GateKeeper service is started, this method is called by the user that should receive credit for work done by the GateKeeper's resource. The user invokes the GateKeeper's `Register` method with a signed, time-stamped message. If the GateKeeper accepts this user's signature, it sends another (signed) message to the Bank. This message contains the original message from the user and information about the resources of the GateKeeper's local machine (i.e. processor speed and disk capacity). The later information is used by the Bank to calculate how much a user is debited for using the GateKeeper's resources. The GateKeeper is able to access the local machine's certificate which it uses to sign its outgoing messages.

In this prototype, signed messages consist of a function invocation on a remote service, associated parameters, a timestamp and the intended destination URI for the message. .NET allows remote methods to be invoked through proxy classes generated from a web service's WSDL. While these proxies appear as objects in the local namespace, invoking methods on them cause SOAP messages to be sent to the appropriate web service. The WSE automates the generation and signing of SOAP

the Microsoft State Service which provides both application (service level) state and session state.

messages given a binary security token, based on a user or resource's X.509 certificate. Using the WSE, the proxies can be configured such that any SOAP message used to invoke a remote method is sent with SOAP headers representing the message's destination (using WS-Routing [13]), a timestamp (using WS-Security [2]) and security information (also using WS-Security). The SOAP security header contains the certificate used to sign the message and a digest of all or part of the message. In this prototype, the entire message is signed because non-repudiation of both the message body and headers is important. The fact that the GateKeeper's registration message contains the original signed message from the user is important because it proves that the designated user did request an association with *this* GateKeeper at a particular time.

The `RunJob` and `WriteFile` method allow access to the GateKeeper machine's processor and disk resources². These methods are called by VO users who must sign the request messages with their certificates. If the GateKeeper recognizes the signature's DN as belonging to one of the current VO members, the method invocation is allowed (see the discussion of the Enforcement service below for why a request from a recognized VO member might also be disallowed). The `RunJob` method takes the name of a windows executable and job arguments and begins execution of that job on the local processor. The GateKeeper can also queue the job and there is a `JobStatus` method that allows the user to discover the job's status. When execution completes, a resource utilization report is sent to the Bank. This report contains the original signed job request from the user and the job's total processor time (in ms) all under the GateKeeper's signature. This prevents rogue GateKeepers from altering the credit/debt of users who did not request use of their resource (although a user executing on a particular resource must trust that the GateKeeper on that machine did not lie in its resource utilization).

Similarly, the `WriteFile` method takes a byte stream and writes it to the local disk. Upon completion, the disk space used (in KB) is reported to the Bank. Note that it is up to the Bank to determine the relative value of utilizations of different resources.

² Note that a running job may use the disk resource as well as the processor resource. Each job runs in its own GateKeeper-assigned directory. The size of that directory when the job completes is added to the resource utilization report.

4.2. The Bank Service

The Bank service contains information on every member user and resource in the VO. For users, this information consists of their Distinguished Name (DN), the DN of the GateKeeper (i.e. of the machine) whose utilization provides them credit, and their current credit/debit for using other resources in the VO. The resource information consists of resource statistics (processor speed and disk size), and well as the resource's DN and the DN of its associated user.

The Bank exposes three methods, `MemberJoin`, `RegisterGateKeeper` and `UtilizationReport`. `MemberJoin` is called by potential VO members to register their intent to join the virtual organization. The DN that signs that `MemberJoin` message is stored by the Bank. The new member has officially joined the VO when the `RegisterGateKeeper` method is called by a GateKeeper service that is registering itself as being associated with the new member. The `UtilizationReport` function is called by GateKeepers after resources are provided to users through either the GateKeeper's `RunJob` or `WriteFile` method. Based on the policy for the VO, the account of the resource requester is debited and the account of the resource provider is credited.

The Bank is also in charge of the Enforcement service. This service implements the enforcement actions specified by the VO's policy. For each new member that joins the prototype VO, the Bank creates a new Enforcer (see below). If the Bank determines that a member is no longer complying with the VO's policy (i.e. one of the VO policy's enforcement conditions has occurred), the Bank sends a (signed) message to the appropriate Enforcer, instructing it to take a particular enforcement action.

4.3. The Enforcement Service

The Enforcement service contains a set of "enforcer objects", each of which can be run in an independent thread to handle enforcement actions as directed by the Bank. The Enforcement service exposes 2 methods, `CreateNewEnforcer` and `PerformEnforcement`, which respond only to messages signed by the Bank. `CreateNewEnforcer` takes a member DN and a list of current VO GateKeepers. A new Enforcer object is created and associated with that DN. Initially, this Enforcer must contact the VO's GateKeepers and direct them to add the new member to their "access permitted" lists.

The `PerformEnforcement` method takes a member DN and an enforcement action number. This number corresponds to the various enforcement actions listed in the VO policy. The Enforcement service will locate the Enforcer object for the designated user and have it begin the appropriate action. Typically, this will involve the Enforcer contacting various GateKeeper's in the VO and directing them to reduce (or eliminate) the services that they are willing to provide to the given user. The Bank can also direct an Enforcer to remove restrictions on a given user if that user causes an enforcement condition to become false. Since this prototype implements a `ygywg` VO, this will happen when the Bank receives enough utilization reports from the GateKeeper that credits the restricted user's account. When that user's credit is raised above threshold, the Bank will direct the appropriate Enforcer to remove any previous restrictions.

One interesting aspect of the Enforcement service's design is that it pushes information about VO members to VO resources. This has two important effects. First, it prevents VO resources from having to poll for new member information. Since the Bank service knows when new information is available, it is natural for it to direct an Enforcer to send out that information. Second, this model allows VO users to communicate directly with VO resources using no central mediating authority. Since each GateKeeper has its own access control list, it can make local decisions about fulfilling resource requests. This provides greater efficiency to users since it does not rely on all requests flowing through a central point, such as the Bank.

However, this model does present a challenge. As the number of resources in the VO increases, the time to push new information to all of them increases. This can lead to some GateKeeper's being "out of sync" with the current VO enforcement state. It can also cause confusion on the part of a recently joined user as to why certain resources are available to them and certain others are not (because they have not yet received the initial message about this new user from the user's new Enforcer). The prototype does not address this issue because highly-synchronized VOs are beyond our current scope. However, the VO needs to characterize the risk of resources consumed by a user between the time that an enforcement action is begun against that user and the time that all appropriate resources have been notified. The "exposure" window can be reduced by having multiple Enforcer objects perform the communication. The number of Enforcers involved in contacting GateKeepers can depend on the severity of the enforcement action. In general, we feel that the benefits of allowing local resource control

decisions are outweighed by the risk of unreturned resource utilization.

4.4. The Prototype's VO policy

The prototype virtual organization uses the policy shown in Figure 1 (expressed in XML).

```
<?xml version='1.0'?>
<VOPolicy>
  <conversion>
    <processor>
      <normalizer>.001</normalizer>
      <units>ms</units>
    </processor>
    <disk>
      <units>KB</units>
    </disk>
    <between>
      <processor>10</processor>
      <disk>1</disk>
    </between>
  </conversion>
  <enforcement>
    <condition>
      <debt>20000</debt>
    </condition>
    <action>
      <mail>member</mail>
      <cc>provider</cc>
    </action>
  </enforcement>
  <enforcement>
    <condition>
      <debt>40000</debt>
    </condition>
    <action>
      <privilege>0</privilege>
    </action>
  </enforcement>
  <initCredit>20000</initCredit>
</VOPolicy>
```

Figure 1. Policy for the ygywg prototype VO

This policy is read by both the Bank and the Enforcement service. The purpose of this work is not to propose a new standard for policy language, but rather to explore concepts involved in systems that must maintain policy across a VO and not just in a local service or site. While we present a policy here to make our ideas more concrete, it is recognized that more work is needed to define the policy syntax and semantics such that all VO members (particularly new members) can interpret it properly³.

³ Incorrect use of the policy by a new member's GateKeeper would result in enforcement actions against that GateKeeper. While the new member would regrettably see reduced service, the rest of the VO members would not be adversely effected by the new GateKeeper's misinterpretation.

The policy is divided into two main sections, the conversion section and the enforcement section. The conversion section describes how the Bank should calculate the cost associated with any particular resource utilization report it receives. The conversion section first contains tags for each resource on which the Bank can receive reports. The units of each resource report are given in the <units> tag. The <normalizer> tag expresses how the costs of various resources in the same class are compared. If a normalizer tag exists, the Bank uses the formula:

$$\text{cost} = \text{usage} * \text{resource_stats} * \text{normalizer}$$

In the case of processor utilization, the usage is multiplied by the processor speed reported to the Bank when the GateKeeper joined the VO (faster processors cost more to use). The <normalizer> tag in the <processor> section causes one cost unit to be equal to 1 ms of run time on a 1 GHz processor. If a <normalizer> does not exist, such as in the case of the <disk> resource, the Bank does not factor in any resource statistics. So, for this VO, the cost of using disk resources does not depend on the size of the disk (though presumably, a VO might want to “charge” more for using smaller or faster disks). The <between> tag denotes the conversion factor between utilizations of resources different types. This policy sets 10 units of processor utilization equal to 1 unit of disk utilization (in effect 10 ms of processor time on a 1 GHz processor is equal to 1 KB of storage). Note that these conversion factors are arbitrarily chosen for this prototype.

The enforcement section describes the enforcement conditions and actions for this VO. Each <enforcement> tag contains 1 conditions and one action. The conditions are expressed in terms of utilization units (see above) of debt. This means that the first condition will be triggered when a user has consumed 20000 more utilization units that have been provided by their associated resource. Similarly, the second condition is triggered at 40000. Again, these particular values are chosen only for illustrative purposes. There are two enforcement actions in this policy. The first is the <mail> action which causes email to be sent to a user with a cc to their associated resource provider. The prototype uses both user and machine certificates issued by a university-wide CA whose policy is to have email addresses as part of a certificate’s DN. The second enforcement action involves reducing a user’s privilege. Users have one of two privileges on the prototype’s GateKeepers (known as 0 and 1). If the user’s privilege is 1, they are allowed to access the resource and if it is 0, they are not. When a user has accumulated more than 40000 units of debt, the Bank will trigger the appropriate Enforcer to contact the VO’s GateKeepers and have them change the user’s

privilege level to 0. The final set of policy tags <initCredit> specify the initial credit that a new VO member user is given.

5. Conclusion

As grid computing evolves, virtual organizations will become more important, have more resources and members and be more difficult to administrate. Virtual organization policy assists the creators and maintainers of a VO by allowing them to specify a set of rules that govern the virtual community. This work describes issues involved in policy specification and enforcement as well as a prototype implementation. This prototype is one of the first to use the WS-Security specification for authentication and non-repudiation which are important in policy-based systems. While this prototype is an important first step, in the future we wish to remove our own custom interfaces to services and replace them with the new OGSi standards for grid services from the Global Grid Forum [9].

6. References

1. Alliance: National Computational Science Alliance. 2002. <http://www2.ncsa.uiuc.edu/About/Alliance/>
2. Atkinson, B., et. al. 2002. Web Services Security (WS-Security). <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp>
3. ATLAS Grid. 2002. <http://www.usatlas.bnl.gov/computing/grid/>
4. Bettini, C., Jajodia, S., Wang, X.S., Wijesekera, D. 2002. Obligation Monitoring in Policy Management. Policy 2002 Workshop.
5. CalTech Virtual Organization Group Manager (“GroupMan”). <http://groupman.sourceforge.net/>
6. Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. 2001. Grid Information Services for Distributed Resource Sharing. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)
7. Dulay, N., Lupu, E., Sloman, M. and Damianou, N. 2001. A Policy Deployment Model for the Ponder Language. Proc. IEEE/IFIP International Symposium on Integrated Network Management (IM’2001)
8. Foster, I., Kesselman, C., Nick, J. and Tuecke, S. 2002. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG, Global Grid Forum.
9. Global Grid Forum. <http://www.ggf.org>

10. GriPhyN: The Grid Physics Network. 2002.
<http://www.griphyn.org/index.php>
11. Keahey, K and Welch, V. 2002. Fine-Grained Authorization for Resource Management in the Grid Environment. Proceedings of Grid2002 Workshop.
12. NASA Information Power Grid. 2002.
<http://www.ipg.nasa.gov/>
13. Nielsen, H. And Thatte, S. 2001. Web Services Routing Protocol (WS-Routing).
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp>
14. NPACI: National Partnership for Advanced Computing Infrastructure. 2002.
<http://www.npaci.edu>
15. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S. 2002. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks.
16. Sundaram, B., and B. Chapman. XML-Based Policy Engine Framework for Usage Policy Management in Grids. Proceedings of the Third International Workshop on Grid Computing (Grid 2002). Baltimore, MD, November 2002.
17. Thompson, M. 2001. Akenti Policy Language.
<http://www-itg.lbl.gov/security/Akenti/Papers/PolicyLanguage.html>
18. Usage Record Working Group. Global Grid Forum. 2002. http://www.gridforum.org/3_SRM/ur.htm
19. Verma, D., S. Sahu, S. Calo, M. Beigi, and I. Chang. A Policy Service for GRID Computing. Proceedings of the Third International Workshop on Grid Computing (Grid 2002). Baltimore, MD, November 2002.
20. Virtual Organization Membership Service.
<http://grid-data-management.web.cern.ch/grid-data-management/security/>