

User Management for Virtual Organizations

*Jiří Denemark³, Michał Jankowski², Luděk Matyska¹,
Norbert Meyer², Miroslav Ruda¹, Paweł Wolniewicz²*

¹ *Institute of Computer Science, Masaryk University
Botanická 68a
602 00 Brno, Czech Republic*

² *Poznań Supercomputing and Networking Center
ul. Noskowskiego 10
61-704 Poznań, Poland*

³ *Faculty of Informatics, Masaryk University
Botanická 68a
602 00 Brno, Czech Republic*



CoreGRID Technical Report
Number TR-0012
November 30, 2005

Institute on Grid Information and Monitoring Services

CoreGRID - Network of Excellence
URL: <http://www.coregrid.net>

Abstract

Scalable and fine-grained Grid authorization requires the move away from grid-mapfile based access control and 1-to-1 mappings to individual OS user accounts. This is recognized and addressed by virtual organization (VO) authorization services, e. g. VOMS/LCAS and CAS. They do, however, not address user OS account management and isolation/sandboxing requirements, such as flexible pooling of accounts while maintaining auditing records. This paper describes some existing systems for user management for VOs and provides a list of requirements for a new user management system which our current research is focused on.

1 Introduction

The main aim of user management system is controlled, secure access to grid resources. Security requires authentication of the user and authorization based on combined security policy from the resource provider and virtual organization of the user. The second important thing is possibility of logging user activities for accounting and auditing and then gathering these data both by the resource provider and virtual organization of the user. From the user point of view, an important feature is single sign-on.

The problem of user management is a non-trivial one in an environment, that includes bulk number of computing resources, data, and hundreds or even thousands of users participating in lots of virtual organizations. The complexity rises from the point of view of time required for administration tasks and automation of these tasks. There are many solutions that attempt to fulfill these basic requirements and solve the mentioned problem, but none of them, according to our best knowledge, solve the problem in complex and satisfactory way.

2 Definitions

Virtual organization (VO) is a set of individuals and/or institutions that allows its members sharing resources in a controlled manner, so that they may collaborate to achieve a shared goal [1].

We assume that virtual organizations may form hierarchies. The hierarchy of VO is useful for user management on the VO side (delegation of administrative burden to sub-organization in case of big organizations) and accounting (sub-organizations may refer to real institutions and departments who are responsible for paying the bills). The hierarchy forms a Directed Acyclic Graph (DAG) where the VOs are vertices and the edges represent relations between them (see [5], sub-organizations are called “groups”).

The user may be a member of many VOs, and in particular, member of a sub-organization is also member of parent organization.

The privileges the organization wants to grant the user, related to the tasks he is supposed to perform, are connected to *user roles*. The roles are defined across the hierarchy of VOs and are managed in independent structure, although the authorities of VOs are responsible for defining roles. One user may have multiple roles and he is responsible to select the required role while accessing the resource (automated selection of a role giving necessary authorization leads to a potential security risk).

Any special rights to resources expressed, e. g., by ACL [4] are called *capabilities*. The capabilities may be used to express any rights to a specific user, e. g., some file is writable only by the owner.

Resource provider (RP) is an abstract entity that owns and offers some resources (e. g. services, CPU, disks, data, etc.) to the grid users.

By the *virtual environment* we understand encapsulation of user jobs in order to give it a limited set of privileges and be able to identify the user and organization on behalf which the job acts. Example implementations are virtual accounts [11, 12, 13], virtual machines [14], and sandboxes [7].

3 Existing Solutions

In this section, we provide a brief description of several systems trying to cope with user management in the context of virtual organizations.

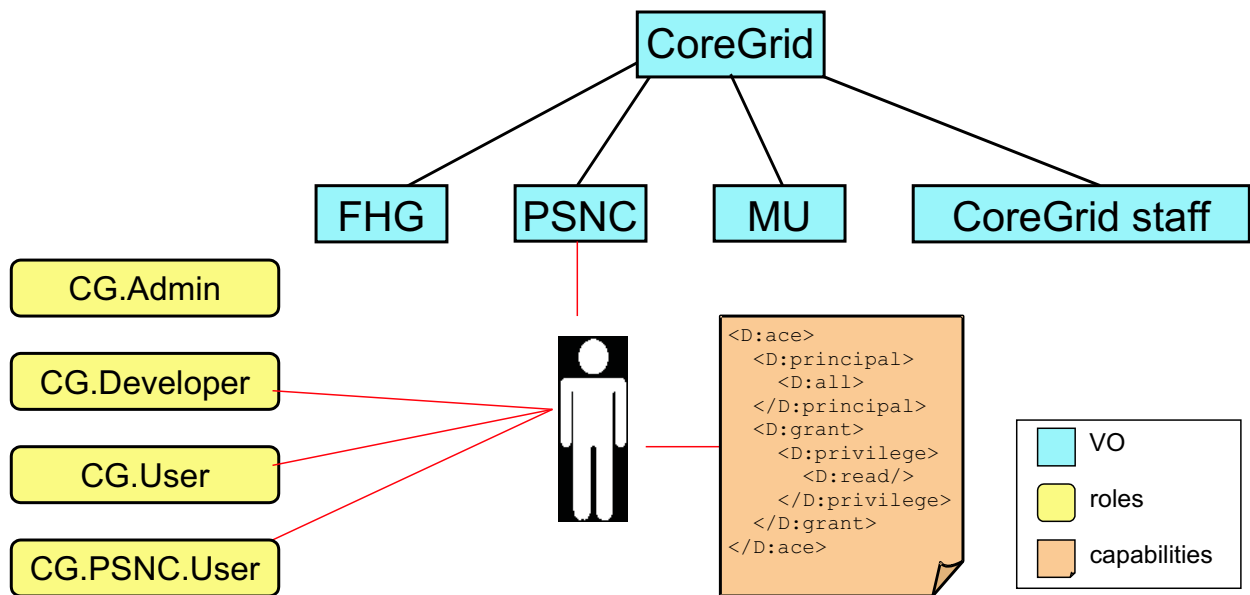


Figure 1: Hierarchy of Virtual Organizations, User Roles and Capabilities

3.1 Perun

Project Perun [15] started as a user management tool for the Czech national Grid encompassing heterogeneous resources from supercomputing centers across Czech Republic. The system itself is not a symmetric component of authorization service. On the contrary, it provides a repository of complex authorization data, as well as tools to manage the data. The data are used to generate configuration of the authorization services themselves (starting from UNIX user accounts through grid-mapfiles to VOMS database). In turn, these services are used to enforce authorization policies. Hence the centralized Perun architecture does not introduce a single point of failure of the whole Grid authorization infrastructure.

Perun makes use of central configuration repository which models an *ideal world*, i. e. how the resources should look like. In this central repository, all the necessary (and possibly very complex) integrity constraints are relatively easy to be enforced. The repository is complemented with a change propagation mechanism which detects the changes, generates consistent configuration snapshots of atomic pieces of managed systems, and tries to deliver them to their final destinations, dealing with resource or network failures appropriately. In this way, the *real world* is forced to follow the ideal one as closely as possible.

The core of the system is completely independent on the structure and semantics of the configuration data, hence the system is easily extensible. In the current deployment the managed configuration items include user accounts on UNIX machines, Kerberos realms, and user home directories on both UNIX filesystems and AFS.

For international Grid projects GridLab and EGEE, Perun was extended to cover additional requirements of these. First, support for binding X509 identities to physical users, and maintenance of a set of trusted certification authorities were added. The data are used for generating *grid-mapfiles* (mapping of X509 certificate subjects to local user names) directly. The files are either propagated together with `/etc/passwd` using standard Perun's mechanism or published on a web site from where they are picked by the managed computers actively.

The EGEE project uses two different authorization services: LCMAPS and VOMS. The current Perun implementation supports both. For LCMAPS it generates and publishes LDIF files which are picked by the service and turned into local *grid-mapfiles*. VOMS is controlled directly via its administrative interface.

3.2 Virtual User System

Virtual User System (VUS) [13] is an extension of the system that runs users' jobs (e. g. scheduling system, Globus Gatekeeper, etc.) and allows running jobs without having a personal user account on a node. The personal accounts are replaced by "virtual" ones, that are mapped to users only for time needed to fully process a job. In this way, it is

no need to maintain user accounts for all users on each computing node, reducing administrative overhead associated with such environment.

There are groups of virtual accounts on each computing node. Each group consists of accounts with different privileges, so the fine grain authorization is achieved by selecting appropriate group by an authorization module. The authentication module is pluggable and may be easily extended or replaced. For example, the authorization decision may be based on VO-membership of the user: the plugin may call a remote VO service to ask about the user or get his VO name from the credential signed by the VO. The VO manager decides who is member of the VO. The grid node administrator (resource provider) decides which VOs should be authorized, defines connection with VO-account group and configures privileges of accounts. The second plugin may check if the user is present in a banned users list, maintained by the (site) administrator. This leaves with RP enough administrative power while part of this power and most of the administrative work is delegated to VO manager. In that way security policies of VO and RP are combined.

Once the user is authorized, he is mapped to a virtual account from the group of accounts suggested by the authorization module. The mapping user-account mechanism assures that only one user is mapped to a particular account at any given time. The history of user-account mappings is stored in a database, so that accounting and tracking user activities are possible.

3.3 VOMS, LCAS and LCMAPS

Virtual Organization Membership Service (VOMS) [4] contains database with information on the user's Virtual Organization and group (sub-organization) membership, roles and capabilities. The service preserves it in a special format—the VOMS credential. The user, before starting a job must acquire the VOMS proxy certificate signed by his VO and valid for limited time. The extra authorization data is placed as a non-critical extension in the proxy, so it is compatible with not VOMS aware services. In order to take an advantages of VOMS data, the Globus Gatekeeper was extended by LCAS and LCMAPS.

Local Center Authorization System (LCAS) is a service used on computing nodes in order to enforce local security policies. The authorization decision is based on user proxy certificate and job description and is made by pluggable authorization modules. VOMS-aware modules use user's VO membership, roles and capabilities for making the decision.

Local Credential Mapping Service (LCMAPS) maps user to local credentials (AFS/Kerberos tokens, UNIX account and group), depending on user proxy certificate and job description. The mapping decision is based on user proxy certificate and job description and is made by pluggable authorization modules.

3.4 Virtual Workspaces, Runtime Environments, Dynamic Virtual Environments

Very interesting work in the area was done by researchers from Argonne National Lab and University of California [6, 7, 8, 9]. They proposed and implemented Workspace Management Service, which allows to run user jobs in virtual environment (see section 2), using different technologies (GRAM Gatekeeper, OGSF, WSRF). The virtual environments are implemented as dynamically created Unix accounts and virtual machines.

These works deal widely with problems connected to job encapsulation and provide some efficiency comparisons of different virtual environment implementations based on tests performed on prototype system and testbed. The authorization issues are addressed closer only by [6], where RSL-based policy language was proposed.

4 System Requirements

4.1 Authentication

The first step in obtaining an access to a remote resource is authentication. From the user point of view, the remote access should be as much as possible simple and similar to the local access. This may be achieved by fulfilling the following requirements [1, 2]:

- Single sign-on—the user must be able to access all the resources after first successful authentication.
- Delegation—the user must be able to allow any (possibly remote) application to run on his behalf, possibly with limited privileges.

- Integration with local security solutions—the resource providers may use different security solutions (e. g. SSH, Kerberos), but the system should both cooperate with these solutions and hide these details from the user.
- User based trust relationships—resources from different providers may be used together by the user without need of any security configurations done amongst these RPs.

The mentioned requirements are fulfilled by Globus Toolkit [16] to a great extent.

4.2 Authorization

The concept of virtual organizations allows for easier decentralization of user management (each VO is responsible for managing some group of users). On the contrary, in the classical solution each computing node must store authorization information locally on each user (e. g. in Globus grid-mapfile), which obviously is not scalable and brings a nightmare of synchronization. On the other hand, the resource provider must have full control on who and how uses his resources. So that, the security policy should be combined from two sources: VO and RP.

The second important issue is fine grained authorization [6], that allows limiting user access rights to specific resources. The authorization decision must depend on privileges granted to the user by VO (role) and possibly overwritten by RP (e. g. RP may want to ban certain users).

The detailed requirements for the authorization are:

- The security policy of VO and RP must be combined. RP delegates some administrative rights to VOs, but the RP must still have the full power on its resources. The RP does all the security configuration on its nodes, which includes mapping grid privileges granted to the user by VO to local privileges.
- Fine grained authorization. The VO certifies user's membership, his role and capabilities. The authorization is based on the triplet VO, role, capabilities [4] and is done on the computing node. The RP policy defines privileges for given pair VO-role and interprets the capabilities. RP policy may limit the privileges in any way, including denying access at all (e. g. RP bans users from the banned list). The virtual environment should be able to enforce the (limited) privileges.
- User's request must be self-contained so that RP does not need to contact any external entity to obtain any information (such as VO, role(s), capabilities) required to authorize the user. This additional information must be stored within the request in an expandable way. Each RP should take into account only the information it understands and ignore all unknown parts of the request.
- The authorization module should be plug-in based in order to allow flexible configuration (use different set of plug-ins with different priorities) and easy integration with existing authorization systems, services or mechanisms.
- Enforcing quotas—in some applications, (e. g. pre-paid systems) it may be required to suspend or delete a job after quotas expiration/overdraw. Related problem is estimation of resource usage before starting the job in order to avoid canceling jobs done in 90%. The quota should be soft or it should be possible to suspend the job for some time (e. g. to give a chance for finishing them after paying some money). Pre-paid service may be important in application-on-demand solutions.

4.3 Encapsulation of Jobs and Results

The system must assure, that two different jobs will not interact in unwanted and unpredictable way, e. g. overwriting each other results. Moreover, it must be possible to identify who and when used specific resources, performed or attempted some actions. This is relevant from the security and accounting point of view. Usually it is not even enough to know the identity of a user, but it is important on whose (which VO) behalf he acted and in which role. On the other hand, in some situations, some cooperation of jobs or access to the same resources may be required. The above may be achieved by proper level of isolation which respects the following:

- The basic model (default configuration) is that all jobs are encapsulated and thus isolated one to another. Final or partial results of a job are not available to other jobs until they are stored to the global space.

- The encapsulation may be weakened in case of workflows of jobs. In that case, the jobs a priori are to be co-operating (e.g. they communicate to each other or share temporary files). It is also reasonable to weaken the encapsulation for performance reasons (preparing several virtual environments for series of small jobs may introduce unnecessary overhead). It should be possible to decide (or specify) if the job should run in an existing environment or if a new one should be created. The system should detect such situation whenever possible and automatically take care of it.
- In complex task, it may be required that the user may have to use multiple roles or identities to gain access to multiple resources. The system should prepare the virtual environment for the user in the way, that it will be possible to separate subtasks performed with different privileges, identities (certificates) and on behalf of different VO (e.g. in order to identify the organization that should be charged with the costs).
- Files stored with some local privileges/IDs/tokens, may be accessed later with different ID (certificate), with different local ID/token [3].
- Access to virtual environment for VO authorities. Usually user's supervisor wants to have rights for controlling and even stopping the job, especially when the user breaks the internal rules or there is a job of higher priority for the VO to be run. Administrator of VO for virtual environment is like Unix root for the real system.
- Access to virtual environment for other users, pointed by the user—"owner" of the environment. It may be required for interactive jobs like steering equipment in a virtual laboratory which needs close collaboration of few users. Another case is when the same "physical" user may want to access the resource with different virtual identities.
- Job encapsulation in a virtual environment should allow limiting access to resources (system calls, network, disk, ...) as much as possible for security reasons. The fact that a resource provider provides computational and other resources to a grid community should not impose new security risks for the RP.
- Possibly, job encapsulation should be done in such a way that from a point of view of RP the encapsulated job is just a black box. Not only the RP does not need to look into the virtual environment where the job is running but the environment should be able to assure a user there is no way of doing that. This will provide a secure execution environment for jobs processing data which no-one except authorized persons should ever see or even know they exist (e.g. medical data).

4.4 Accounting and Logging Facilities

Any production grid, especially commercial one, needs accounting feature. Before issuing any bills, however, the accounting data must be collected (possibly from several locations) and tied to users and VOs.

From the security point of view, it is important to track user activities (e.g. by analyzing system logs) in order to detect any rule-breaking. It must be possible to identify the user who has performed a particular action.

Both of the above require proper encapsulation of jobs (as described in the previous section) and storing history of user-virtual environment mappings.

The detailed requirements are:

- It must be possible to identify the user and context (VO, role, capabilities, time) of any action that is subject to local logging or accounting. Thus it should be possible to fetch all the accounting and logging information from the standard system mechanisms (such as e.g. Unix accounting). In many application it should be sufficient amount of data.
- The system should be capable of storing non standard accounting data. This is especially useful when the node provides access to non-standard resources (e.g. equipment in a virtual laboratory). The data may be put by any external application through special API.
- The accounting data must be easily accessible for all parties involved in a particular action. Resource provider wants to access all information connected with resources it provided to a grid. Virtual organization would like to see all information on resources used by its members so that VO can make internal decisions according to that information. And the end user, who submitted some jobs, would like to access all information on resources consumed by those jobs.

4.5 Other Requirements

- It should be possible to combine “classical” and “virtual” user accounts in some system. User may have a local account on some system and may need to access it from a Grid job instead of using only virtual account assigned dynamically.
- The system architecture must be ready for lightweight virtual organizations. We can expect that in the future the virtual organizations will be very dynamically created and removed after short time (VO created just for a single project). We must assure that this high dynamics will not introduce much administrative burden or computational overhead.
- Granting rights to the “long-lived” resources (e. g. files) for users that changed certificates (e. g. certificate expires, the user gets a new one and needs access to the previously created files). Possibly the CAs should track the history of issued certificates and associated physical users (i. e. proper *identity* management is necessary).
- The virtual environment should not be limited to one implementation and existing implementations should be deployed using common interface to other modules.
- The architecture should be modular and flexible. It should give a chance for easy integration with existing solutions and standards. The modularity embraces plug-in based authorization, replaceable virtual environment module.
- Automatic registering of new users, issuing certificates and any special security considerations connected with this (application-on-demand solutions).

5 Proposed Solution

We realized, while analyzing existing solutions, that there are number of tools that provide for at least part of the functionality mentioned in section 4, however none addresses all the issues. These tools are widely used in number of projects and some of them become kind of standard, so it seems to be reasonable to be compatible with them. Moreover, it makes no sense to implement from scratch an existing functionality. So we propose to put them into pluggable framework, that will combine the features gaining the synergy effect.

The other important design assumption is being concordant to the existing standards and trends in the area of grid computing, especially webservice (WS) approach. WS-Stateful Resource [10] technology seems to be especially promising for our purpose, as it allows for easy modeling virtual environment and managing its life cycle.

We propose webservice responsible for managing virtual environments (Virtual Environment Management Service, Fig. 2), especially creating and destroying them as well as running jobs in them. In the background, the service collects data on the virtual environments concerning time of creation and destruction, users mapped to the environment, accounting and logging information. These data will be available to different players on the scene like the users, managers of VOs, resource owners etc. via the second webservice called Virtual Environment Information Service, Fig. 3.

5.1 Virtual Environment Management Service

The Virtual Environment Management Service consists of two main modules: authorization module and virtual environment module.

The authorization module performs authentication first. This may be based on existing Globus GSI. The authorization is done by querying set of authorization plugins. The set is configurable, so that the administrator may tune local authorization policy to the real Grid needs and abilities. We plan to implement plugins for most often used authorization mechanisms and services like grid-mapfile (possibly dynamically updated by Perun), CAS, VOMS. The plugins play role of policy decision points (PDP). The authorization decision itself may be done locally (in case of push model of authorization which we prefer) or by querying remote authorization service (pull model—not preferred, but possible). The plugins should not only answer the question if the user is allowed to perform the specified service action, but also give some hints for the parameters of the virtual environment, e. g. grid-mapfile plugin will tell the (virtual) account name, VOMS will expose ACL that will help with creation of virtual machine with specific limitation.

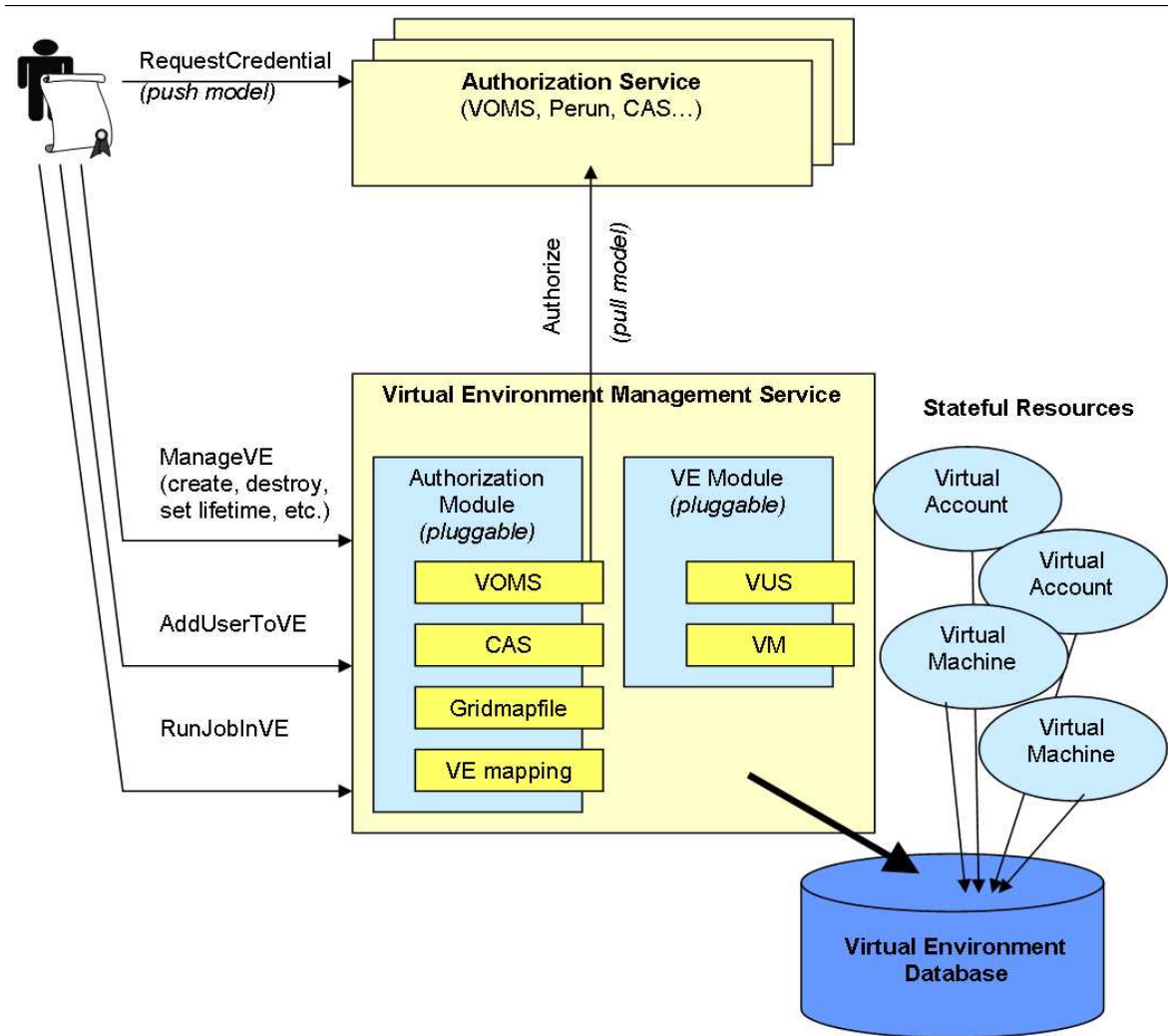


Figure 2: Virtual Environment Management Service

The special authorization plugin is VE mapping, introduced in order to satisfy requirements about loosing isolation level of the VE. This plugin will check a special Access Control List for the VE. While the VE is created, the list will contain one user—owner (creator) of the VE—with full rights to the VE. Then, the owner can add users to the VE either with limited (e.g. only read access to files or job execution) or full (including VE life cycle management) privileges. Note, that the user—owner of VE—takes much responsibility for the added users, because while loosing isolation level we also loose requirements for identifying user and context. The extent of this loosing depends on VE implementation; namely it is impossible to distinguish users if they run jobs in the same virtual account, but it is possible to distinguish them if they run jobs on different accounts of the same virtual machine.

Virtual environment module is responsible for creation, deletion and communication with virtual environments, implemented as stateful resources. The module is also pluggable, so it is possible to use different implementations of VE. It is planned to implement Virtual User System plugin and at least one plugin for virtual machine. The module records all its relevant operations (especially like VE creation and deletion) to the Virtual Environment Database.

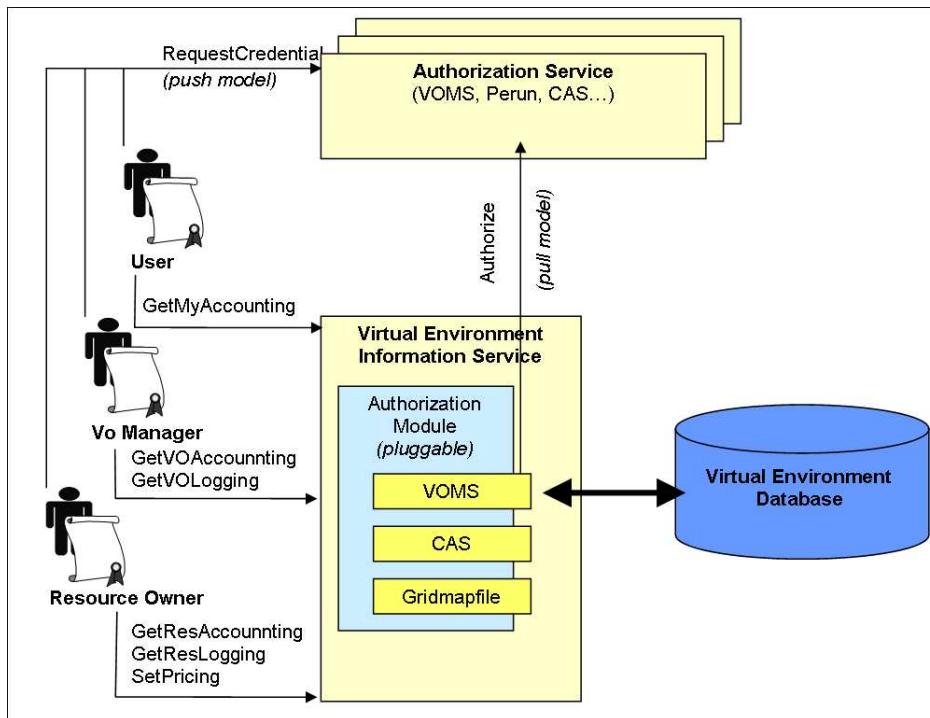


Figure 3: Virtual Environment Information Service

5.2 Virtual Environment Database

The records of VE operations together with the standard system logs and accounting data will provide complete information on user actions and resource usage. However these two sources must be combined and the result put to the database. This might be implemented as database trigger that collects the logging and accounting data periodically or while the VE is destroyed. It must be also possible to put some non-standard data to the database (e. g. time of usage of laboratory equipment connected to the system).

For billing purposes accounting information must be connected with prices. The price is computed depending on the pricing policy of the resource owner. In the simplest model, the database contains a dynamic price list and the current price is put together with accounting record.

5.3 Virtual Environment Information Service

This service is a front-end for the Virtual Environment Database. The access to the data must be authorized and depends on the user role:

- *Common users*, who have run jobs—they should have rights to read the accounting data referring to themselves (e. g. in order to control their budget).
- *Managers of virtual organizations*—should be able to read logging and accounting data of all VO members, to have full control on budget and behavior of the users.
- *Owners of resources*—should be able to access all the data connected to the resource. In the simplest case, the resource is the whole local node or cluster on which the VE runs and the owner is the system administrator. In more sophisticated case the resources may be differentiated and there can be more owners (e. g. usage of some software, owed by some local user may be subject for the accounting). The owners also should have right to modify the pricing policy.

This service will require authorization module with set of plugins similar to the Virtual Environment Management Service, but they must take a bit different decision, mainly based on the user role mentioned above.

5.4 Example Use Case Scenario

1. The user acquires VOMS credential from the VOMS authorization service.
2. The user requests creating virtual environment to the Virtual Environment Management Service.
3. The authorization module of Virtual Environment Management Service checks the VOMS credential and passes the authorization information to the VE module.
4. The VUS plugin maps the user to an account from the group of accounts mapped to the VO of the user. This is recorded in the database.
5. User starts a job. This is recorded in the database.
6. The user gets the output data and requests destroying the VE.
7. The VE is destroyed by the service. This is recorded in the database.
8. The destroying triggers in the Virtual Environment Database process of collecting accounting data from the standard system accounting.
9. The VO manager requests information on accounting of the user to the Virtual Environment Information Service (he presents the VOMS credential).
10. The service authorizes the manager and queries the database. The result is sent to the manager.

6 Acknowledgment

This work has been supported by the CESNET Research Intent (MSM6383917201) and by the EU CoreGRID NoE (FP6-004265).

References

- [1] I.Foster, C.Kesselman, S.Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, 15(3), 2001.
- [2] R.Butler, D.Engert, I.Foster, C.Kesselman, S.Tuecke, J.Volmer, V.Welch, Design and Deployment of a National-Scale Authentication Infrastructure, IEE Computer, 33(12) 2000
- [3] EGEE JRA3, DJRA3.2: Site Access Control Architecture, <https://edms.cern.ch/document/523948/>, 2004-12-31.
- [4] R.Alfieri, R.Cecchini, V.Ciaschini, L.Dell'Agello, A.Frohner, A.Gianoli, K.Lentey, F.Spataro, VOMS: an Authorization System for Virtual Organizations, 1st European Across Grids Conference, Santiago de Compostela, February 13-14, 2003.
- [5] R.Alfieri, R.Cecchini, V.Ciaschini, L.dell'Angelo, A.Gianoli, F.Spataro, F.Bonnassieux, P.Broadfoot, G.Lowe, L.Cornwall, J.Jensen, D.Kelsey, A.Frohner, D.L.Groep, W.Som de Cerff, M.Steenbakkens, G.Venekamp, D.Kouril, A.McNab, O.Mulmo, M.Silander, J.Hahkala, K.Lorentey Managing Dynamic User Communities in a Grid of Autonomous Resources, Computing in High Energy and Nuclear Physics, La Jolla, California, 24-28 March 2003.
- [6] K.Keahey, V.Welch, S.Lang, B.Liu, S.Meder Fine-Grain Authorization Policies in the GRID: Design and Implementation 1st International Workshop on Middleware for Grid Computing, 2003.
- [7] K.Keahey, K Doering, I.Foster, From Sandbox to Playground: Dynamic Virtual Environments in the Grid, 5th International Workshop in Grid Computing (Grid 2004), Pittsburgh, PA, November 2004
- [8] K.Keahey, M.Ripeanu, K.Doering, Dynamic Creation and Management of Runtime Environments in the Grid, Workshop on Designing and Building Web Services (GGF 9), Chicago, IL, October, 2003.

- [9] K.Keahey, I.Foster, T.Freeman, X.Zhang, D.Garlon Virtual Workspaces in the Grid, Europar 2005, Pisa, Italy, August, 2005.
- [10] I.Foster, J.Frey, S.Graham, S.Tuecke, K.Czajkowski, D.Ferguson, F.Leymann, M.Nally, I.Sedukhin, D.Snelling, T.Storey, W.Vambenepe, S.Weerawarana Modeling Stateful Resources with Web Services, version 1.1 <http://www-128.ibm.com/developerworks/library/specification/ws-resource/>, March 2004.
- [11] W. Dymaczewski, N. Meyer, M. Stroiki, P. Wolniewicz, Virtual Users Account System for Distributed Batch Processing, in: P. Sloot, M. Bubak, A. Hoekstra, B. Hertzberger (Eds.): HPCN 1999, LNCS 1593, Springer-Verlag, 1999.
- [12] M.Kupczyk, M.Lawenda, N.Meyer, P.Wolniewicz: Using Virtual User Account System for Managing Users Account in Polish National Cluster, HPCN,Amsterdam, June 2001.
- [13] M.Jankowski, P.Wolniewicz, N.Meyer Virtual User System for Globus based grids, Cracow '04 Grid Workshop Proceedings, December 2004.
- [14] T.Lindholm, F.Yellin, The Java Virtual Machine Specification (2nd Edition), Addison-Wesley, 1999.
- [15] Ales Křenek and Zora Sebastianová. Perun – Fault-Tolerant Management of Grid Resources, Cracow '04 Grid Workshop Proceedings, December 2004.
- [16] Globus Toolkit Version 4: Software for Service-Oriented Systems. I. Foster. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.