On Usage Control in Data Grids

Federico Stagni[1], Alvaro Arenas[2], Benjamin Aziz[2], and Fabio Martinelli[3]

[1] INFN sez. di Ferrara, via Saragat 1 - 44100 Ferrara, Italy
stagni@fe.infn.it
[2] e-Science Centre, STFC Rutherford Appleton Laboratory, Oxfordshire, UK
{A.E.Arenas,B.Aziz}@rl.ac.uk
[3] IIT-CNR, via G. Moruzzi 1 - 56123 Pisa, Italy
Fabio.Martinelli@iit.cnr.it

Abstract. This paper reasons on usage control in Data Grids. We adapt the UCON$_{abc}$ usage control framework for the case of distributed systems with multiple authoritative points. We call it the distributed usage control model. Then, we present an architecture implementing such model. In doing so, we use the functional components of the current Grids. Finally, we show a simple way for controlling the policy granularity using Semantic Grid technologies for the specification of policy subjects and objects.

1 Introduction

Data Grids [22] are an innovative technology taking advantage of existing computer science concepts in file systems, database systems and Grid computing. A Data Grid provides services that help users discover, transfer, and manipulate large datasets stored in distributed repositories and create and manage copies of these datasets. As a minimum, a Data Grid provides two basic functionalities: a high-performance reliable data transfer mechanism and a scalable replica discovery and management mechanism. However, as in any resource sharing environment, robust and rigorous treatment of data security in a Data Grid is vital. Moreover, since data is being shared over multiple administrative domains over the Grid, continuous monitoring and control of the data access is required.

At the present time, the majority of Data Grid middlewares and tools are growing behind some specific needs, mainly HEP (High Energy Physics) experiments. HEP applications produce and consume a considerably high amount of data with heavy impact on the bandwith, but probably they don't need a high security system, because the main purpose of this activities is to be fast. At the same time, other Grid middlewares grow for chemicals or bioinformatics necessities, with different, tighter, security requirements.

A growing number of researchers and Virtual Organizations (VOs) will born. They will use Grids, peer-to-peer systems, or whatever distributed paradigm will be in place that could help with their computing needs. These VOs may pose new security requirements. Just to make the simplest example, in the next generation of Grids file sharing, a user will want to give access to his/her files only to a limited set of people, identified by some kind of property. To do this, there's the need for a high control over who is authorized to view or modify the data [8].

E. Ferrari et al. (Eds.): TM 2009, IFIP AICT 300, pp. 99–116, 2009.

Every Grid application may have a specific set of security requirements, and a Grid middleware should be capable to deal with a vast number of those. Different Grid applications should be able to determine the way the Grid guarantees data integrity and confidentiality. Different Grid authentication and authorization capabilities need to be in place. The solution is to conceive a really flexible system, with no explicit bindings with a specific application. To deal with these requirements, we apply usage control methodologies in a distributed security model, and apply it to a Data Grid abstraction.

This paper studies usage control techniques for Data Grids. Usage control extends traditional access control by controlling data access as well as usage [17,15]. Recently there has been a fresh interest in applying usage control to Grid systems [10,25]. We develop here a usage control model suitable for multi-authoritative distributed systems. We base this model on the UCON_{abc} model proposed by Park and Sandhu [14]. The main contribution of the paper is a Data Grid usage control architecture using the functional components of the current Grids, as presented by the Open Grid Forum (OGF) group on Grid authorization. We also consider the advantages of using Semantic Grid techologies for the specification of UCON subjects and objects for controlling the policy granularity.

The rest of the paper is structured as follows. Section 2 introduces an abstraction of Data Grids and some terminology. In Section 3, we give some background on UCON and introduce the distributed usage control model. Section 4 shows the proposed Data Grid usage control architecture. In Section 5 we reason how to take advantage of Semantic Grid technologies for controlling the policy granularity. Finally, Section 6 discusses related work, and Section 7 concludes the paper and highlights directions for future works.

2 An Abstraction of Data Grids

A distributed system may contain a variety of data resources. These resources may use different data models to structure the data, different physical media to store it, different software systems to manage it, different schema to describe it, and different protocols and interfaces to access it. The data may be stored locally or remotely; may be unique or replicated; may be materialized or derived on demand. Different levels of virtualizations over these data resources should be provided. Virtualizations provide abstract views that hide these distinctions and allow the data resources to be manipulated without regard to their nature. The Data Grid abstraction we provide here helps us with future reasonings.

In a Data Grid there are two kinds of resources to be managed: *Grid Data* and *Grid Storage Space*. A *Grid Data* (GD) is any kind of data that can be located, transferred, replicated and manipulated: client services should be able to access a dispersed GD, independently from its physical location, through a *Data Grid Management System* (DGMS) [12]. A DGMS is a software system used to manage Data Grids through the use of multiple abstraction mechanisms that hide the complexity of distributed data and heterogeneous resources. This naming capability allows users to refer to specific data resources in a physical storage system using a high level logical identifier. A *Grid Storage Space* (GSS) is a storage space shared between multiple VOs, and managed by a Grid *Storage Element* (SE). An SE (e.g. the Storage Resource Manager [7]) is an

interface to mass storage systems, providing a uniform control interface and enabling the Grid to efficiently use the storage.

It is not necessary for a GD to be stored in a GSS only, while a GSS may also contain data that cannot be relocated, viz. are not GD. We are are not interested in the security implications of non-GD data.

DGMS implementations should follow the OGF recommendations for providing implementation guidelines and standards to implement GD location independence. Data resources have to be recognized by name without any location information. The Open Grid Services Architecture (OGSA) work on data architecture [1] identifies a scheme with the following three levels of naming:

- **Human-Oriented Name** (HON)**:** Based on a naming scheme that is designed to be easily interpreted by humans, viz. human-readable and human-parsable. The HONs are user friendly high-level identifiers by which the users find the actual locations of their files. The same data resource could be addressed by various HONs by different users, similarly to the concept of alias. A number of HONs can be mapped to a single *Abstract Name*.
- **Abstract Name** (AN)**:** A persistent name suitable for machine processing that does not necessarily contain location information. ANs are given to each data managed by a DGMS. An AN is a unique identity to hide the data replication: the same AN can correspond to different replicas.
- **Address:** Specifies the location of a data resource. An address provides an abstraction of the data namespace living into a storage resource to allow different data access paths. Each replica has its own address and it specifies implicitly which storage resource needs to be contacted to extract the data.

Figure 1 shows a simplified logical view of a Data Grid. We distinguish two kinds of data accesses: *(i)* clients (e.g. Grid users) access a GD knowing just the HON by

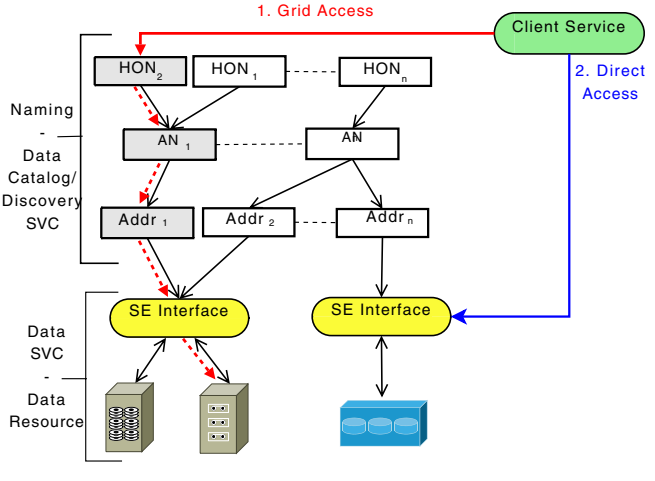

Fig. 1. A logical view of a Data Grid

performing what we call a *Grid access*, and *(ii)* access GD and non-GD data directly on the SE when the address is known, thus performing what we call a *direct access*.

3 A Distributed Usage Control Model

In this section we define a usage control model for Grids and distributed systems. The model can use the UCON$_{abc}$ usage control model for the specification of usage control policies. We chose UCON$_{abc}$ because of its high capabilities, as will be clear by reading below. Then, we'll apply this model to the Data Grid abstraction of Section 2.

3.1 UCON$_{abc}$

The main novelty of the UCON model lies in the fact that subjects and objects may have attributes that are mutable thereby facilitating the continuity of the decision making and policy enforcement processes. Additionally, while decisions in access control models are usually based on *authorizations* only, the UCON model introduces two other decision factors, namely *obligations* and *conditions*. All of these features render the UCON model attractive for specifying security policies in Data Grids, especially considering the plethora of various security needs coming from the different Data Grid applications.

The UCON model comprises the following elements:

- *Subjects, Objects and Rights*: the subject is the entity that exercises rights, i.e. that executes usage operations on objects. An object, instead, is an entity that is accessed by subjects through access operations. Rights are the privileges that subjects can exercise on objects. Traditional access control systems view rights as static concepts, for instance access matrices, which do not change over time or have a slow rate of change. Instead, UCON determines the existence of a right dynamically, whenever a subject attempts to use and exercise a right on some object. Hence, if the same subject accesses the same object several times, the UCON policy could grant the subject different access rights each time based on changing attributes of the subject and/or the object.
- *Attributes*: both subjects and objects have attributes. These attributes can be *mutable*, i.e. they can change over time, or *immutable*, i.e. they are constant over time. An example of a mutable attribute is the number of times that a subject accesses an object, whereas an immutable is a subject's or an object's identity. Conditions can use attributes representing the system status which are not under the UCON service control.
- *Predicates*: predicates are logical statements about the subjects' and objects' attributes and the requested right. Predicates can be either *authorization*, *obligation* or *condition* predicates or any combination of these. Authorization predicates express a set of rules that determine whether to grant the requested right or not. The authorization predicates could exploit both attributes of the subject and of the object. The evaluation of the predicates can be performed before and during the execution of an action. Obligations are UCON decision factors that are used to verify whether subjects have satisfied, or continuously satisfy, some mandatory requirements before (during) an usage. Finally, conditions are environmental or system-oriented

decision factors that do not depend on subjects or objects. Conditions are evaluated at runtime when the subject attempts to perform the usage, before or during an action [26].

$UCON_{abc}$ is a family of models with several parameters. The presence of Authorizations (A), oBligations (B) and Conditions (C), pre- and on-going decisions, as well as the mutability of attributes (immutable (0), preUpdate (1), onUpdate (2), postUpdate (3)) are the factors to be considered. For example, a $PreA_0$ policy is an pre-authorization policy with no attributes update, while an OnB_3 is a on-obligation policy with a postUpdate of one or more attributes, and so on. The various UCON models differ in the presence of attribute updates and in the sequentiality of the operations. Therefore, an enforcing mechanism for UCON policies should be able to enforce not only the single operations, but the sequence these operations are invoked. The different actions that subjects and system can perform in the UCON model relate to the different phases of an object's usage.

Given that the triple `(s,o,r)` represents the subject `s` requesting the right `r` for accessing the object `o`, we consider the following set of actions, which we borrowed from [26]: *(i)* `TryAccess(s,o,r)`: performed by subject `s` when performing a new access request `(s,o,r)`, *(ii)* `PermitAccess(s,o,r)`: performed by the system when granting the access request `(s,o,r)`, *(iii)* `DenyAccess(s,o,r)`: performed by the system when rejecting the access request `(s,o,r)`, *(iv)* the operation `RevokeAccess(s,o,r)` is performed by the system when revoking an ongoing access `(s o,r)`, *(v)* `EndAccess(s,o,r)`: performed by a subject `s` when ending an access `(s,o,r)`, and *(vi)* `AttributeUpdate(s,o,r)`: performed by the system to update a subject or an object attribute when performing an access request `(s,o,r)`.

All the policies pertaining to the UCON authorization, obligation and condition core models are defined for positive permissions: if there is no policy to enable the permission according to the attribute values, then the usage is denied by default. This is sometimes called the closed system assumption, whereby no policy is specified to deny an access in a system.

3.2 The Distributed Usage Control Model

Up to now, there's no existing security model that can cope with the inner nature of Grids. In a distributed system like a Grid, there may be small to larger number of different resources, each one controlled by a different policy officer. Each policy officer is a *Source of Authority* (SoA) for an *authoritative point*, viz. authoritative sources of authorizations and usage control. When a client service is requesting the permission to access a single remote resource, a number of policies maintained by different SoAs may have to be evaluated. This requirement was historically advocated by the Globus and EGEE (Enabling Grids for E-science) security teams [20] and, up to now, there is no existing Grid usage control framework coping with this requirement. Therefore, the challenge for controlling the resource usage in Grids and distributed systems is knowing which are the authoritative points involved in a usage request. The model we propose in this Section can deal with such requirement.

Within the model we propose, that we call *Distributed Usage Control Model* (D-UCM), policy officers could impose the evaluation of local policies. We say that a single

usage decision comes from the evaluation of a *workflow* of local usage control steps. For example, when the workflow of a complete usage control is made of three separate usage control steps, each one of the three must be satisfied. If one of the usage control steps can't be satisfied, the entire usage is not permitted.

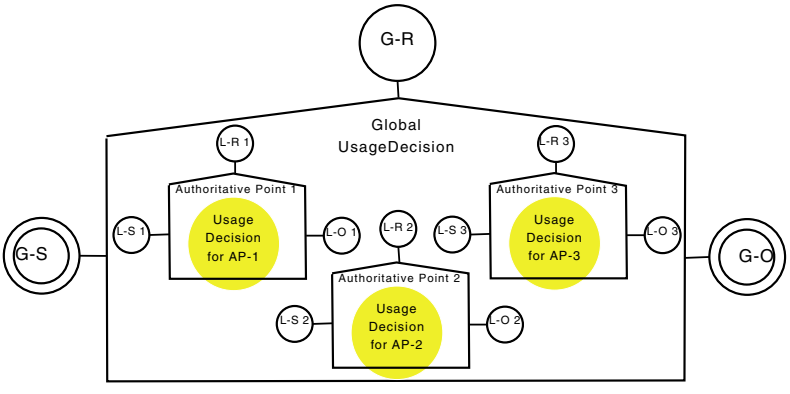

Fig. 2. The Distributed Usage Control Model

Figure 2 shows a pictorial overview of D-UCM. Within this Figure, we show that three distict authoritative points each impose the evaluation of a local usage decision (L-UD) step. Each step have to be satisfied for the enforcing of a global usage decision (G-UD). A central workflow orchestrator, with responsibility for the G-UD, is needed. The evaluation of a L-UD step is seen as an atomic action. The model doesn't pose any constraint neither on the way authoritative points enforce usage control steps, nor on the nature of the security policies that have to be evaluated to reach a L-UD. For example, a L-UD may require the evaluation of a vast number of distributed and cuncurrent policies, but all this machinery is under the responsibility of the local Source of Authority (SoA).

A G-UD is based on a global subject (G-S), a global object (G-O) and a requested global right (G-R). To reach a L-UD, each SoA encode G-S, G-O and G-R respectively in a local subject (L-S), local object (L-O) and local right (L-R). The relation between the global and local subjects, objects and rights is dependent from the application using the model.

4 Usage Control in Data Grids

Policy-based security mechanisms adopt an almost standard terminology when defining authorisation architectures, which distinguishes between different kinds of *Policy Points*. Their definition has strong connection with traditional access control techniques. In this paper, we continue using the same terminology when drawing a distributed usage control architecture for Data Grids. In doing so, we use functional components defined by the OGSA-Authz GWD-I draft architecture for a Grid service provider authorisation service middleware [3].

In the OGSA work, great attention is put on *credentials*, defined as attribute assertions digitally signed by the issuer (i.e. a security token) so that it can be cryptographically validated. Credentials can be issued by the Credential Issuing Services (CISs) of an Identity Provider or an Attribute Authority (e.g. the Virtual Organization Membership Service (VOMS) [21]). Credentials can then be validated by a Credential Validation Service (CVS), that return the valid attributes of the subject. A Policy Decision Point (PDP) is the component responsible for returning an authorization decision given the user's access request and the user's valid attributes. The Policy Enforcement Point (PEP) enforces the results returned from a policy engine (normally a PDP). The Context Handler (CH) is responsible for handling the communications between PEPs, CVSs and PDPs. The interactions between these functional components can be constructed in four different ways, according to whether the credentials and the authorization decisions are pulled or pushed. For example, Figure 3 shows the case where an access requestor (a Grid User) pushes his/her credentials to a PEP. Then, after the CH obtained valid attributes from the CVS, a PDP is interrogated for an authorization decision, which in the end is returned to the PEP.

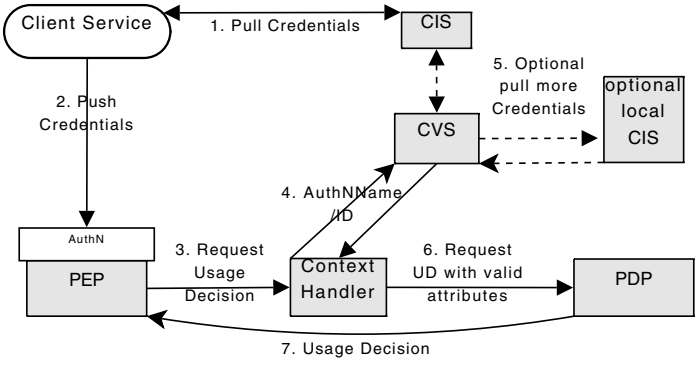

Fig. 3. OGSA functional components

4.1 A Data Grid Usage Control Architecture

We now apply the *distributed usage control model* (D-UCM) of Section 3.2 to the Data Grid. The reason we chose UCON_{abc} as a policy model is because it encompasses traditional access control models, and does not pose constraint on the degree/level of granularity of usage control, ranging from storage space level to individual data access restrictions.

We now consider the terminology introduced in Section 2. In a Data Grid, GDs (Grid Data) are stored (and transferred and replicated) in GSSs (Grid Storage Speces) by the SEs (Storage Elements). A client performing an access to a GD should be authorized to access the data itself, and to use the GSS. Therefore, the policies of the single steps should be written by those policy officers which are SoA for the GDs (e.g. VO admins

or simply VO participants), and by those policy officers which are SoA for the GSSs (e.g. SE admins). Therefore, we identified a couple of authoritative points, which are DGMS and SEs. A Complete usage control in Data Grid then follows a two-steps workflow. From now on, we refer to each of these steps as *data usage control* (D-UC) and *storage usage control* (S-UC). Each step corresponds to the enforcing of (at least) a UCON_{abc} policy.

We now pose some constraints on the relation between G-S, G-O, G-R and L-S, L-O and L-R of D-UC and S-UC. Each single L-S represents the G-S as it is recognized by respectively the DGMS and the SE. Similarly, the L-R represents the G-R as it is recognized by the DGMS and by the SE. The object of the D-UC is the unique identifier of a GD, i.e. the *abstract name*. The object of S-UC is, instead, the GSS itself. By doing this neat separation between the objects of D-UC and S-UC, we highlight the role of the authoritative points. Moreover, by doing this separation, the policies of the different steps will never overlap. Figure 4 shows this two-step usage control.

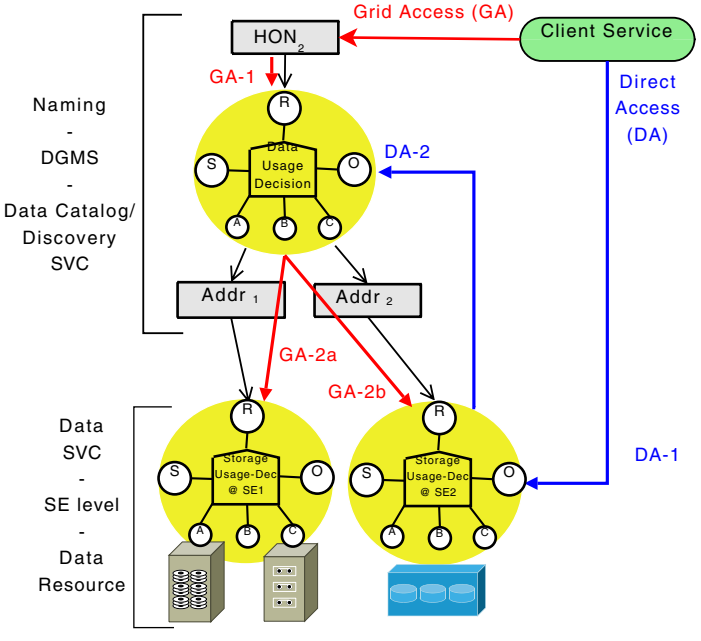

Fig. 4. The two-steps Data Grid usage control

There are many differences between existing security models for Grids and the one we are proposing, but the most apparent one can be seen in Figure 5. The Figure shows security models for Data Grids as seen in [6] with the D-UCM for Data Grids (on the right). The main difference stands with the usage of two (UCON) PDPs, one for each usage control step.

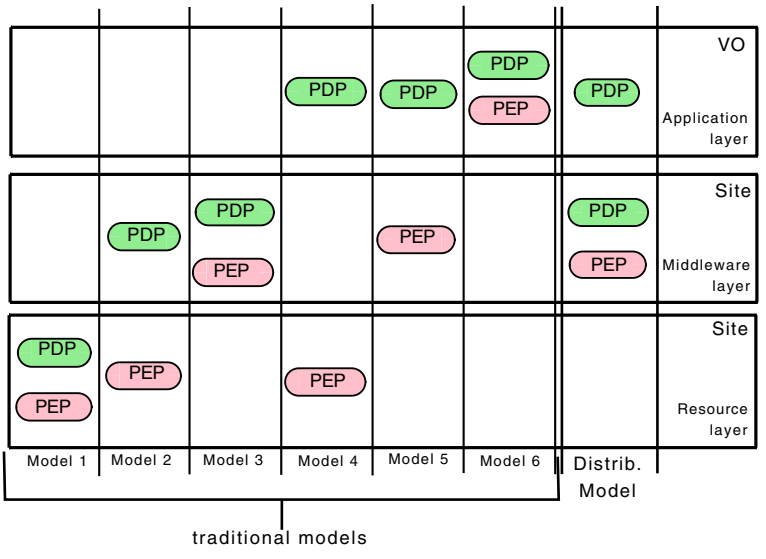

Fig. 5. Comparison of security models for Data Grids (inspired from [6])

Figure 6 shows a Data Grid security architecture implementing the D-UCM on Data Grids. Each usage control step uses the authorization functional components defined by OGSA. A *Client Service* is an access requestor (normally, a Grid User) that pushes the credentials obtained from a VO CIS either to a DGMS (when performing a Grid access) or to a SE (when perfoming a Direct Access). DGMS and SE are clients to a *super-PEP* software element, which communicate with the CHs located at the DGMS and SEs. Each CH obtains valid attributes from the CVS. Then, the local UCON PDP is interrogated for an authorization decision. A UCON PDP should be capable to interpret, i.e. enforce, policies pertaining to the UCON_{abc} usage control framework. The UCON PDP is responsible for returning an usage decision to the super-PEP, given the user's usage request (i.e. the right requested), the user's valid attributes, the object's valid attributes, and the satisfaction of authorizations, obligations and conditions predicates. From a UCON point of view, valid attributes released by a CVS are examples of *immutable* (persistent) attributes.

The *super-PEP* is the software element responsible for performing both the usage control steps requested. Among the possible solutions for this element, a centralized service or a collaborative one. For instance, one could consider POLPA [11], a policy language suitable for expressing sequence of actions as well as conjunctions and disjunctions of such sequences. These policies could be useful to *orchestrate* other usage control steps in a workflow (as well as to model single access actions in a usage control step). A possible initial solution in this line of thought is envisaged in [2]. Due to the fact that a super-PEP may be located at DGMS or at SE level, we consider it as a mobile agent.

A complex UCON PDP should be able to evaluate policies where the predicates are statements about the subjects' and objects' attributes. Five sub-components make up the UCON PDP:

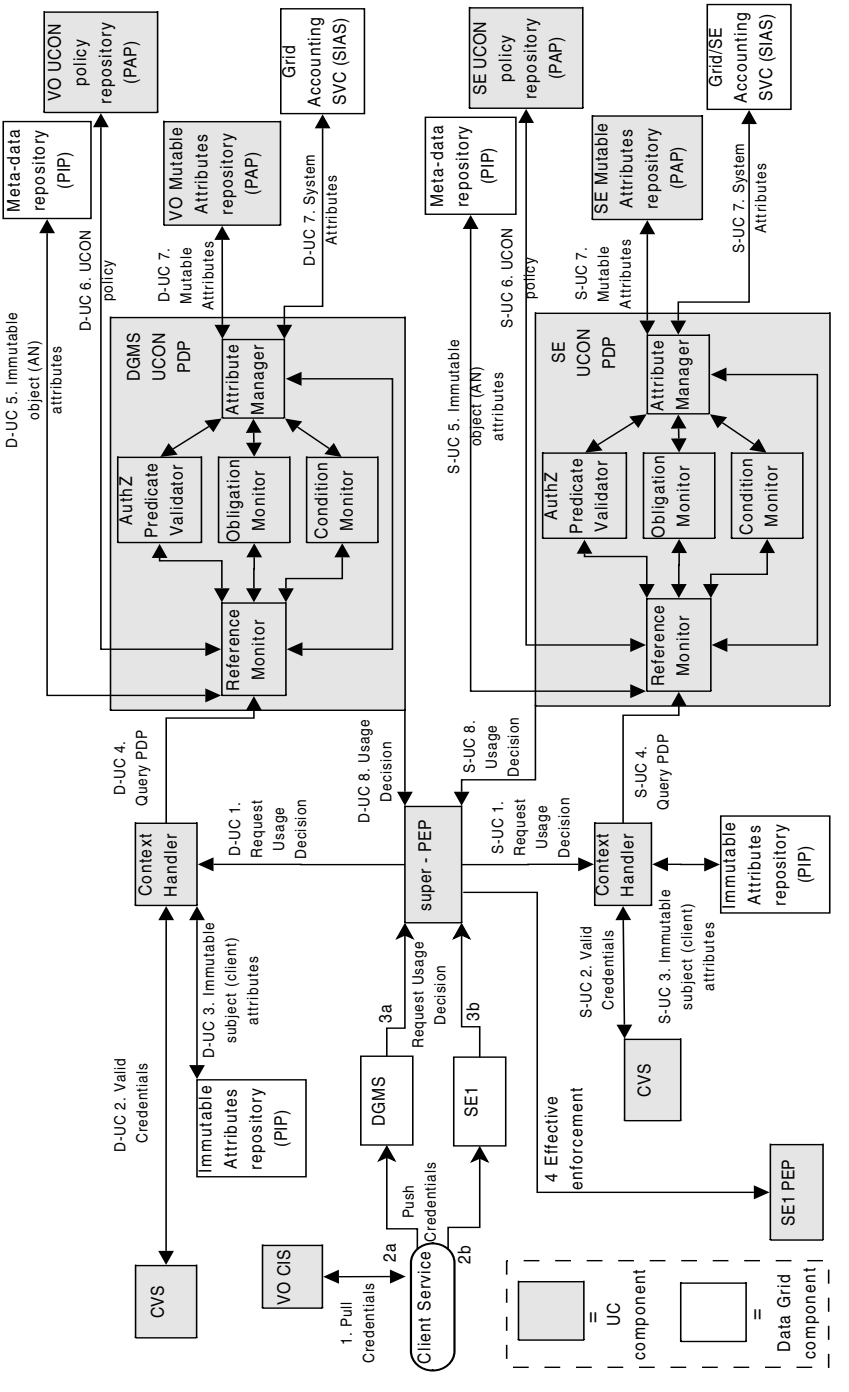

Fig. 6. Data-Grid usage control architecture

- the **Reference Monitor** (RM) is a gateway for all the usage decisions; it can receive `TryAccess` and `EndAccess` invocations, and is responsible for issuing the `PermitAccess`, `DenyAccess` or `RevokeAccess` operations;
- the **Authorization Predicate Validator** (PV) takes care of validating the authorization policy predicates; it can be perform the `AuthzPredicateValidation` operation;
- the **Obligation Monitor** (OM) checks if subject fulfilled the obligations; it can be perform the `ObligationsSat` operation;
- the **Condition Monitor** (CM) takes care of validating the condition policy predicates; it can be perform the `CondsPredicateValidation` operation;
- the **Attribute Manager** (AM) updates the UCON mutable attributes and return their values; it can perform the `AttributeUpdate` operation.

External components are needed to supply the UCON PDP with the needed information: *(i)* an *UCON policy repository* provides the PDP with the UCON policies to be evaluated, *(ii)* a *meta-data repository* provides the PDP with the optional immutable object attributes, *(iii)* a *mutable attributes repository* stores the UCON mutable attributes of the subjects and objects, and *(iv)* the *Grid/SE Accounting SVC* is a System Information/Accounting Service (SIAS) acting as a source for system attributes. For an access, the PDP collects the immutable subject and object attributes, as well as search for the UCON policies to be enforced. The policy is selected using the the UCON subject and object requested. Mutable subject and object attributes, as well as system attributes, are pulled by the PDP from the mutable attribute repository, and from the Grid accounting service.

For what regards the *data usage control* step, the rights are defined at the level of the *abstract name*. Thus, we apply the following restrictions: *(i)* an **UCON subject** is represented by a DGMS user ID, which is the way the access requestor Grid user ID is recognized by the DGMS; *(ii)* an **UCON object** is represented by the *abstract name* requested by the DGMS user ID; *(iii)* an **UCON right** always follows in one of the fundamental rights categories, which are *view* (read) and *modify* (write), possibly augmented with *creation* and *deletion*; *(iv)* **subject attributes** are mutable or persistent security descriptors of the *Client Services* (e.g. the number of data accessed); *(v)* **object attributes** are mutable or persistent security description of the *abstract name* (e.g. the *privacy* level, or the maximum number of contemporary access);

Insted, since the *storage usage control* step defines rights at the address level, we apply the following restrictions: *(i)* an **UCON subject** is an SE user ID, which is the way the access requestor Grid user ID is recognized by the SE; *(ii)* an **UCON object** is the GSS where the GD is located; *(iii)* an **UCON right** depends from the SE interface implementation in use; *(iv)* **subject attributes** are security descriptors of the *Client Services*; *(v)* **object attributes** are security descriptors of the GSS;

4.2 Architecture Analysis

Main Pros

- The whole architecture is **modular**, **flexible**, and presents a **high capability** level. A number of policy officers are capable of specifying policies pertaining to a vast

number core models, and these policies will never overlap. Moreover, each SoA maintain a local authority over its resources, and there's no need for policy syncronization.

Main Cons

- **Complexity.** The proposed architecture has a high degree of complexity. We are aware of the fact that *Complexity is the worst enemy of security.* [1]. There are reasons for such complexity, and simplification possibilities. All the software elements composing the UCON PDP have been recognized as requirements for enforcing UCON_{abc} policies. To do so, we used notions that are partially extracted from the KAOS requirement engineering methodology to produce an abstract specification of all the UCON PDP architectural elements and operations. Such work is partially available in [18]). We also demonstrated that such specification is capable to enforce all the UCON_{abc} types of policies, as they are formally specified in [26].

  An overall simplification is possible: since UCON is a family of core models, simpler UCON PDPs would enforce not all, but a number of UCON core models. For example, the *Obligation Monitor* component is not necessary if there are no needs for enforcing UCON_{b} policies.
- **Performance and Trust.** Other big problems may be represented by the performance of an implementation, and by the trust relationships between the sites, but since right now there's not a single complete implementation of the architecture, we leave this problem to future works on the topic.

Issues

- **Policy strategy.** Near the end of Section 3, we mentioned that an enforcement mechanism for UCON policies should be able to enforce not only the single operations, but the sequence these operations are invoked. In order for an UCON PDP to be an enforcement mechanism for all the UCON core policy models, a way to encode the policy *strategy* (i.e. the sequentiality of the operations) is needed. A possibility lies in the use of an operational policy language like the already cited PoLPA, where the policy specification itself encodes the strategy. Otherwise, an external scheduler can be used for the particular UCON_{abc} sub-model to which the policy pertains.
- **Obligations.** Checking the obligations satisfaction is still an issue. An introductory work on usage control obligations can be found in [16]. We don't plan to solve such issue within this paper.

We believe this concrete architecture can be of real use for implementors and developers.

5 Usage Control in Semantic Grids

As stated in [4], the Semantic Grid is an extension of the Grid in which rich resource metadata is exposed and handled explicitly, and shared and managed via Grid

[1] Bruce Schneier, Crypto-Gram newsletter, March 2000.

protocols. The layering of an explicit semantic infrastructure over the Grid infrastructure potentially leads to increased interoperability and greater flexibility.

In the near future, data on the order of hundreds of petabytes will be spread in multiple storage systems worldwide dispersed in, potentially, billions of replicated data items. The creation, definition and enforcement of usage control policies may represent an issue in terms of management, scalability, governability and consistency. For example, in current hierarchical file systems, access control is made specifying the authorizations on every one of billions of files. If usage and access control techniques are to be really useful in a large pervasive environment, they should be able to solve the scalability and governability problems presented by the more traditional access control models, such as Identity Based Access Control (IBAC) — normally implemented using Access Control Lists (ACLs) — or even the more flexible Role Based Access Control (RBAC) [5]. In the implementations of traditional access control models, when an authorization policy changes for a specific user or role, the security manager must implement the adjustment in every entry involved, potentially all. These factor may generate a policy explosion phenomenon. What's needed is a mechanism for keeping under control the policy granularity. A simple solution lies in the semantic binding assertions regarding Grid users and resources, as exposed in a Semantic Grid. UCON subjects and objects may be semantic concepts extracted from those VO ontologies or scientific model ontologies used in the Semantic Grid.

Before going any further, we make a clear distinction between semantic attributes and UCON attributes. Semantic attributes can globally describe users, data and resources properties, but are not meant to be security attributes. Instead, the UCON attributes define only subjects' and objects' security properties, and for many of them there is no need to be known outside the usage control service. In a Semantic Grid, following the terminology introduced in [4], each *Grid Entity* is associated to a *Knowledge Entity* (KE) through a *Semantic Binding*. KEs are special types of Grid Entities that represent or could operate with some form of knowledge. Examples of KEs are ontologies, rules, knowledge bases or even free text descriptions that encapsulate knowledge that can be shared. Semantic Bindings are the entities that come into existence to represent the association of a Grid Entity with one or more KE.

A semantic-aware UCON PDP is depicted in Figure 7, and is obviously much similar to the one presented with Figure 6. In a Semantic Grid, the client service (i.e. the Grid User) and the data to be accessed (e.g. the abstract name managed by the DGMS) are represented by a KE. For what concerns the DGMS, the metadata repository can be used to store the KE of the abstract names. Even if in Semantic Grids specific Grid Users will keep asking to access specific Grid Data, a semantic-aware PDP would search for applicable policies using the multiple fields of the KEs of both the Grid user and the resource to be accessed. In this way, two or more policies could be applicable for a single access request, thus generating more than a single policy control for a single access request. When no policy is applicable, the access is denied. When multiple UCON Pre$\{ABC\}$ policies are to be evaluated, even if just one is satisfied, then the access has to be permitted. When multiple UCON On$\{AB\}$ policies are to be evaluated, even if just one is no more satisfied, then the access will be revoked.

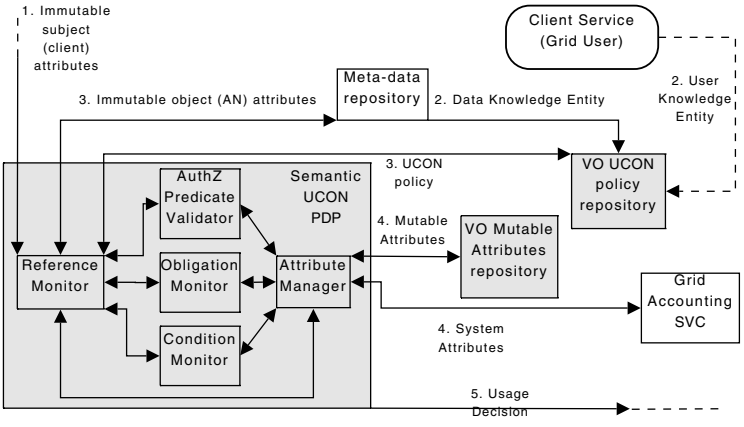

Fig. 7. A semantic-aware UCON PDP for Semantic Grids

Example of KEs representing the Grid Entity *Grid User* and the GD are shown in Figure 8. A semantic-aware DGMS could associate a data KE like this one to each of the managed *abstract names*. The Grid User KE graph is inspired from [4], while GD graph has been derived from the CCLRC scientific metadata model [19]. These examples are not meant to be complete. Each Grid User is simply described through the use of three fields: the *Institution* he/she is affiliated with, the *Investigation* he/she takes part in, and the *Job or Role* he/she is doing as part of the *Institution*. Instead, each GD is described not only by the *Type* (e.g. file, or stream), but also by the *Program* of work, the supported *Study*, and by an *Investigation*.

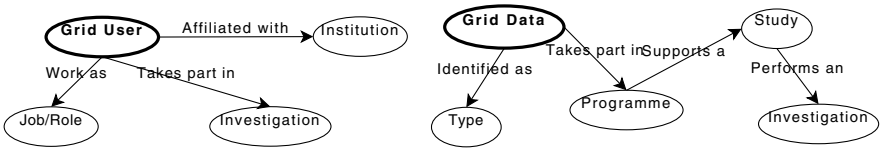

Fig. 8. An example for a Grid user and a GD Knowledge Entity

A security administrator can control the policy granularity using the semantic fields shown in Figure 8 for the definition of collective policies, like the following simple PreA_0 policy (written in POLPA, where "." represents sequence of actions):

```
1 TryAccess(Institution:STFC, Study:ISIS, read).
2 PredicateValidation([]).
3 PermitAccess(Institution:STFC, Study:ISIS, read).
4 EndAccess(Institution:STFC, Study:ISIS, read).
```

This policy states that each User associated with the Institution `STFC` can read those GD pertaining to the `ISIS` study. UCON attributes can be associated to these UCON subjects and objects.

With this simple approach, it's easy to realize a fuzzy security [24] for Grids. The possibility to control the policy granularity, and thus to avoid the policy explosion is of particular interest for those VOs that consider the specification of a per-user, per-role or per-data policies a useless effort. High Energy Physics VOs usually fall in this category.

6 UCON Implementations

We are not aware of specific usage control frameworks for Data Grids, although there are already some running implementations for other scenarios.

In [10], Martinelli and Mori provide a model for usage control for computational Grids for the Globus Toolkit, following Sandhu's UCON model. The prototype implements the standard PEP-PDP architecture, and the PoLPA policy language is used to encode UCON policies. The PEP has been integrated within the application execution environment to monitor the accesses to the local resources performed by the applications executed on behalf of remote GRID users. The PDP gets the security policy from a repository, and builds its internal data structures for the policy representation. The PDP is invoked by the PEP every time the subject attempts to access a resource. It exploits its representation and determine whether the access should be allowed or not, returning to the PEP `permit` and `deny` invocations. The PDP continuously evaluates a set of given authorizations, conditions and obligations while an access is in progress, and it could invoke the PEP to terminate it through a `revoke` action. The architecture comprises the managers for attributes, conditions and obligations. The *Condition Manager* is invoked by the PDP every time the security policy requires the evaluation of a condition. The *Attribute Manager* is in charge of retrieving and updating the value of attributes. The *Obligation Manager* monitors the execution of obligations. Martinelli and Mori focussed on single GRID computational services. We argue that the adaptation of UCON to Data Grid poses a greater number of issues to be solved. This paper highlighted a number of them.

In [25], Zhang *et al* propose a UCON prototype implementation. The security architecture leverages a centralized attribute repository in each VO and a usage monitor in each Resource Provider (RP) for attribute management. The policies are specified with the eXtensible Access Control Markup Language (XACML) [13], which, as recognized by the same authors, seems suffers of several limitations to exactly encode UCON policies. Both PDP and PEP are located on the RP side. For an access, the PDP collects the subject, object and system attributes, and makes the usage control decision, which is enforced by the PEP. The immutable subject attributes are pushed to the PDP by the requesting subject. This prototype has not been applied to an actual (Data) Grid security architecture, like the OGSA one.

Due to increasing number of kernel-level attacks The protection of the kernel integrity is one of the most essential security goal in building a trustworthy operation system. An approach based on UCON model for Linux kernel protection was proposed at [23].

Pioneer works, specifying usage control requirements with mobile and ubiquitous computing application, were presented at [9].

Even if these prototypes should be considered when implementing a Data Grid usage control architecture, none of them consider the inner multi-authoritative nature of Data

Grids and their specific issues. We plan to implement our architecture and check its feasibility and performances for real applications by starting from these experiences.

7 Conclusion and Future Work

Different Grid applications running on the same middleware may need different security levels. A middleware security service should be modular and flexible, in order to accomodate disparate Grid applications authorization requirements. We believe that usage control techiniques, as presented in this paper, are a step toward the right direction. We proposed a usage control model for Grids and distributed systems that uses a *workflow* of usage control steps. Each step implements a distinct usage control through the enforcemente of at least a UCON_{abc} policy. In a Data Grid, a complete usage control is performed with two separate steps. Then, we presented a flexible distributed usage control architecture for Data Grids with a strong reference to the OGSA work on Grid authorization architecture. We also showed a simple way of using the Semantic DataGrids KEs for controlling the policy granularity, thus avoiding the policy explosion phenomenon.

We consider this paper as a step toward an integrated usage control framework for Data Grids. We believe that many of the ideas presented here can be adapted for the case of computational Grids and distributed systems alike. Regardless of it, there is still issues to be solved. Some of them have been highlighted in Section 4.2. We are currently analysing deployed policy languages and authorisation mechanisms in order to determinate their capacity to implement UCON_{abc} policies as presented here, looking at the possibility of extending one of the already developed implementations. The final goal is to either propose a new implementation, or extensions to the already developed ones. Works in these directions have already started.

Acknowledgements. This work is partially founded by the EU CoreGRID project, contract No. 004265, and the EU GridTRUST project, contract No. 033827.

References

1. Antonioletti, M., Berry, D., Chervenak, A., Kunszt, P., Luniewski, A., Laws, S., Morgan, M.: Ogsa data architecture v0.6.6. Technical report, Open Grid Forum (2007), `http://forge.gridforum.org/sf/go/doc13635?nav=1`
2. Aziz, B., Arenas, A., Martinelli, F., Matteucci, I., Mori, P.: Controlling usage in business process workflows through fine-grained security policies. In: Springer (ed.) 5th International Conference on Trust, Privacy& Security in Digital Business (2008)
3. Chadwick, D.: Functional components of grid service provider authorisation service middleware. Technical report, Open Grid Forum (2008), `http://forge.gridforum.org/sf/go/doc15171?nav=1`
4. Corcho, Ó., Alper, P., Kotsiopoulos, I., Missier, P., Bechhofer, S., Goble, C.A.: An overview of s-ogsa: A reference semantic grid architecture. J. Web Sem. 4(2), 102–115 (2006)

5. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed nist standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) (3), 224–274 (2001)
6. Frohner, A., Kunszt, P.Z., Brito da Rocha, R., Laure, E.: Security of distributed data management. Technical Report EGEE-TR-2006-003. EGEE-TR-2006-DATASEC (2006)
7. Group, T.S.R.M.W.: An internet attribute certificate profile for authorization (2008), http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.pdf http://sdm.lbl.gov/srm-wg/doc/SRM.v.2.2.pdf
8. HealthGrid: Healthgrid white paper. Technical Report HealthGrid-White_Paper-Draft_v.1.1-5, HealthGrid (2004)
9. Hilty, M., Pretschner, A., Schaefer, C., Walter, T.: Usage control requirements in mobile and ubiquitous computing applications. In: ICSNC 2006: Proceedings of the International Conference on Systems and Networks Communication, p. 27. IEEE Computer Society, Los Alamitos (2006)
10. Martinelli, F., Mori, P.: A Model for Usage Control in GRID systems. In: Grid-STP 2007, International Conference on Security, Trust and Privacy in Grid Systems. IEEE Computer Society, Los Alamitos (2007)
11. Martinelli, F., Mori, P., Vaccarelli, A.: Towards continuous usage control on grid computational services. In: ICAS/ICNS, p. 82 (2005)
12. Moore, R., Jagatheesan, A., Rajasekar, A., Wan, M., Schroeder, W.: Data Grid Management Systems. In: Proceedings of the 21st IEEE/NASA Conference on Mass Storage Systems and Technologies, Maryland, USA (2004)
13. OASIS: Oasis extensible access control markup language (xacml) tc (2005), http://www.oasis-open.org/committees/xacml
14. Park, J., Sandhu, R.: The UCON_{abc} Usage Control Model. ACM Transactions on Information and System Security 7(1), 128–174 (2004)
15. Pretschner, A., Hilty, M., Basin, D.: Distributed usage control. Communications of the ACM (2006)
16. Pretschner, A., Massacci, F., Hilty, M.: Usage control in service-oriented architectures. In: Lambrinoudakis, C., Pernul, G., Tjoa, A.M. (eds.) TrustBus 2007. LNCS, vol. 4657, pp. 83–93. Springer, Heidelberg (2007)
17. Sandhu, R.S., Park, J.: Usage control: A vision for next generation access control. In: Gorodetsky, V., Popyack, L.J., Skormin, V.A. (eds.) MMM-ACNS 2003. LNCS, vol. 2776, pp. 17–31. Springer, Heidelberg (2003)
18. Stagni, F., Arenas, A.E., Aziz, B.: On usage control in data grids. Technical Report TR-0154, Institute on Knowledge and Data Management, CoreGRID - Network of Excellence (2008)
19. Sufi, S., Matthews, B.M.: The cclrc scientific metadata model: a metadata model for the exploitation of scientific studies and associated data. In: Knowledge and Data Management in Grids (2005), http://epubs.cclrc.ac.uk/work-details?w=34195
20. team, E.J.: Egee global security architecture for web and legacy services. deliverable EGEE-JRA3-TEC-487004-DJRA3.1-v1-1, EGEE JRA3 (2004)
21. Venturi, V., Stagni, F., Gianoli, A., Ceccanti, A., Ciaschini, V.: Virtual organization management across middleware boundaries. In: E-SCIENCE 2007: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, pp. 545–552. IEEE Computer Society, Washington (2007), http://dx.doi.org/10.1109/E-SCIENCE.2007.84
22. Venugopal, S., Buyya, R., Ramamohanarao, K.: A taxonomy of data grids for distributed data sharing, management, and processing. ACM Comput. Surv. 38(1), 3 (2006), http://dx.doi.acm.org/10.1145/1132952.1132955

23. Xu, M., Jiang, X., Sandhu, R., Zhang, X.: Towards a vmm-based usage control framework for os kernel integrity protection. In: SACMAT 2007: Proceedings of the 12th ACM symposium on Access control models and technologies, pp. 71–80. ACM, New York (2007), http://doi.acm.org/10.1145/1266840.1266852
24. Yao, D.: An ad hoc trust inference model for flexible and controlled information sharing. In: Security and Management, pp. 555–561 (2008)
25. Zhang, X., Nakae, M., Covington, M.J., Sandhu, R.: Toward a usage-based security framework for collaborative computing systems. ACM Trans. Inf. Syst. Secur. 11(1), 1–36 (2008), http://doi.acm.org/10.1145/1330295.1330298
26. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal Model and Policy Specification of Usage Control. ACM Transactions on Information and System Security 8(4), 351–387 (2005), http://doi.acm.org/10.1145/1108906.1108908