
Juan M. Marín Pérez, Jorge Bernal Bernabé, Jose M. Alcaraz Calero, Felix J. García Clemente, Gregorio Martínez Pérez, Antonio F. Gómez Skarmeta

Abstract

There are a few issues that still need to be covered regarding security in the Grid area. One of them is authorization where there exist good solutions to define, manage and enforce authorization rules and policies in Grid scenarios. However, these solutions usually do not provide Grid administrators with semantic-aware components closer to the particular Grid domain and easing different administration tasks such as rule conflict detection or resolution. This paper defines a proposal based on Semantic Web to define, manage and enforce security rules in a Grid scenario. These semantic-aware rules help the administrator to create higher-level definitions with more expressiveness. These rules also permit performing added-value tasks such as conflict detection and resolution, which can be of interest in medium and large scale scenarios where different administrators define the authorization rules that should be followed before accessing a resource in the Grid. The proposed solution has been also tested providing some reasonable response times in the authorization decision process.

Keywords: Globus Security, Authorization, Semantic Web, Policy Conflict Analysis

1. Introduction

The management of a Grid implies dealing with multiple security aspects regarding authentication, authorization and communication protection, as well as managing user credentials and group membership information. The configuration and management of these security aspects may become a complex and error-prone task, i.e. the Grid administrator has to deploy a security policy by means of the proper configuration of security components, and moreover, manage this configuration. In most cases, a Grid administrator receives a high-level security policy that must be previously refined before achieving the final security configuration that should be applied to the target Grid elements. A security policy that could appear with a simple textual form like “if a service permits to perform query operations to user A and user B is located in the same laboratory as user A, then the service must also permit to perform query operations to user B” may imply a complex task for the administrator. Moreover, the administrator is limited to the parameters supported by the security scheme that could make it impossible to deploy that policy. This managing complexity is also incremented due to the domain heterogeneity between the organizations which share resources on the Grid and which usually have their own access policy for their resources. For instance, in the previous example users A and B may belong to different organizations.

*Corresponding Author

Email addresses: juanmanuel@um.es (Juan M. Marín Pérez), jorgebernal@um.es (Jorge Bernal Bernabé), jmalcaraz@um.es (Jose M. Alcaraz Calero), fgarcia@um.es (Felix J. García Clemente), gregorio@um.es (Gregorio Martínez Pérez), skarmeta@um.es (Antonio F. Gómez Skarmeta)

Preprint submitted to Future Generation Computer Systems March 18, 2010
An advanced and sophisticated scheme is necessary to allow the definition and validation of high-level policies, providing high expressiveness in the language for policy definition and taking into account complex domain information. It is also necessary to make the management tasks easier, to facilitate the policy deployment to the administrator and to cope with integration issues caused by the domain heterogeneity of the organizations participating in the Grid.

Current solutions like Shibboleth [1], VOMS [2] or PERMIS [3][4] provide a set of syntax constructions in order to enable administrators to express authorization policies. This set of constructions makes up the expressiveness provided by the language. Sometimes, there is a gap between the policy the administrator has in mind and the policy he can describe using the language constructions. In this situation, the administrator is usually able to refine the original policy, but resulting in complex policy definitions with lots of rules or, in the worst case, he may even be unable to enforce the policy in the system. Besides, these approaches are focused on describing security policies and they are not able to cope with the semantics related to the elements they are protecting. For example, a simple policy rule like “Only administrators can read files of the /root folder” entails including the semantics related to the structure of folders and files in order to correctly manage the protection of these resources. This capability of expressing domain semantics is rarely supported by current solutions. Moreover, these approaches do not properly deal with policy conflicts, providing detection capabilities to avoid undesirable or even security risky situations which may arise as a result of errors in complex policy definitions.

This proposal presents an authorization system for Grid security that relies on Semantic Web [5] technologies, achieving a higher expressiveness for policy definitions to reduce the gap between the abstract policy and what the administrator can specify using the language constructions. It provides the capability to describe the semantics of the resources which are to be protected, as well as extending these definitions to fit particular domain requirements. The policy execution engine allows for automated reasoning and enables support for advanced capabilities like policy conflict detection and resolution tasks. The heterogeneity of Grid systems has been also taken into account and the definition of multiple organization domains and policies is supported.

A concrete Grid infrastructure has been used to drive the design of the authorization system in order to provide a real solution. The Globus Toolkit [6] is an open source software toolkit for building Grid systems and applications. It is currently the reference framework for Grid computing and it has been used as base platform for the solution design. Our approach is based on the advantages provided by Semantic Web such as languages to represent knowledge and semantics, standardization and interoperability, reasoning and inference capabilities, and information retrieval, searching and manipulation. A proof of concept implementation has been developed and performance and statistics are also covered.

This paper is structured as follows. First, Section 2 introduces the main ideas behind the usage of Semantic Web for authorization policy representation. Section 3 presents the proposed authorization architecture and its integration within the Globus Toolkit. Details about authorization policy management and the reasoning process are provided in Section 4 and some implementation issues and statistical results are exposed in Section 5. Finally, this paper concludes with the related work in Section 6, and some remarks and statements of direction in Section 7.

2. Semantic Web representation for authorization policies

In order to take policy-based authorization decisions, a representation of policy rules should be provided. Moreover, since policy rules are based on domain concepts, a representation of such a domain is also necessary. These representations should be generic and flexible enough to be able to fulfil different domain modelling needs to cope with the heterogeneity of Grid systems. In this proposal, ontology languages are used to tackle these representations, since they endow with high expressiveness to model concepts and semantics for different domains. These languages provide constructors to define ontologies, where an ontology is defined as a set of axioms which provide a formal representation of a given domain, containing the schemas and vocabularies that define domain concepts, their properties and relationships among them, as well as instances of these concepts and relationships, representing a specific situation in the domain according to those schemas.

The Web Ontology Language 2 (OWL 2) [7] is a W3C standard which enables the specification of ontologies, defining class hierarchies and their relationships, associated properties and cardinality restrictions. Different kinds of OWL 2 ontologies can be distinguished, attending to the provided expressiveness and its computational complexity. In this context, the term expressiveness refers to the set of structural elements or constructors that can be employed in the
axioms for the ontology description. Concretely, three different profiles [8] are defined for OWL 2: OWL 2 EL, OWL 2 QL and OWL 2 RL. OWL 2 EL is designed for ontologies with very large number of properties and classes, mainly providing existential quantification. OWL 2 QL is useful for ontologies with a large number of instances, where query answering is the most important reasoning task. These two profiles are targeted for particular applications, providing high performance reasoning algorithms for them, but limiting their expressiveness. In turn, OWL 2 RL is aimed at providing scalable reasoning, without sacrificing too much expressive power. This last profile has been selected to model the domain description in our solution since it provides a wide variety of constructors to describe knowledge whereas it can be handled by reasoners implemented by means of rule-based engines, which offer good performance and scalability.

This representation allows the inclusion of semantics in the domain description, providing with constructors to define property semantics like inverseOf, transitiveProperty or reflexiveProperty; constructors to define object semantics like equivalentClass, disjointWith or unionOf; as well as cardinality and existential restrictions like minCardinality, allValuesFrom or someValuesFrom; among other features. Moreover, the domain description may be composed of different ontologies which are significant for its representation. For example, the administrator can select an ontology based on CIM, such as [9], for applications, systems, networks, devices, and other technology-focused management domains, or SOUPA ontology [10] for representing ubiquitous and pervasive applications, or also SWRC ontology [11] for modelling entities of research communities such as persons, organizations, publications and their relationships.

Models defined using OWL 2 can be enriched by means of the Semantic Web Rule Language (SWRL) [12]. This language is used to represent rules on the Semantic Web and it extends OWL 2 in order to provide a way to express conditional knowledge. The language itself is not decidable, but a syntactic restriction called DL-Safe context [13] can be applied in order to restore the decidability. The combination of SWRL and OWL 2 can be used to express not only the authorization policy, but also more deductive processes on the system which may be relevant for authorization decisions. For instance, this combination enables the definition of rules to provide semantics related to the resources which are to be protected like “every file in the domain whose name ends with / is a directory” that makes the system to consider these files as directories for authorization purposes. Hence, the OWL 2 RL profile together with the SWRL DL-Safe have been selected to carry out the semantic-aware approach to Grid authorization exposed in this paper. The OWL 2 RL profile and the SWRL with DL-Safe restriction will be indistinctly referred as OWL and SWRL henceforth.

The combination of OWL ontologies and SWRL to specify policy rules offers the advantage of allowing automated reasoning. This is carried out by what is called a reasoner, referring to a specific piece of software which performs reasoning processes. These processes constitute a remarkable added value of the usage of Semantic Web technologies, since they are able to infer new knowledge, that is, deriving additional information not explicitly specified in the ontology. Moreover, they also perform a formal validation and verification of the domain constraints which are specified in the ontology to assure they are fulfilled.

The representation of the policy and domain information used in our authorization system is based on these Semantic Web concepts and technologies, providing the following set of features as added value for our policy language:

- Separation between domain description and policy description. Our approach separates the concepts that are necessary to describe the domain to be protected and the rules which use such concepts to create policies expressing the desired security for the administered services. The separation of domain description and policy description permits us to manage both specifications individually using different techniques.

- Reasoning capabilities about domain descriptions. The management representations are done in the form of an ontology, allowing reasoning processes to check constraints and query the information.

- Reasoning capabilities about policy description. Policy rules are created in the form of horn-like rules, enabling reasoning capabilities for policy conflict analysis techniques.

- Interoperability. The use of standards and ontological representation eases the interpretation and integration of the information. This enables advanced capabilities like concept alignment between heterogeneous domains, allowing the combination of concepts from the different domains of the organizations which are part of the Grid.
Some of the concepts introduced in this section will be further explained and used including some exemplifications in the following sections of this paper, with the aim of making them more clear to the reader.

3. Authorization architecture for Grid services

Globus A&A (Authentication and Authorization) architecture supports authentication and delegation through the use of X.509 certificates and public keys, and an authorization framework that can process restricted delegation and can take the identity of intermediates into account. This authorization framework allows for a variety of authorization mechanisms, including a grid-map file access control list, an access control list defined by a service and access to an authorization service via the Security Assertion Markup Language (SAML) [14]. While the toolkit provides some implementations for these authorization mechanisms, the framework is pluggable and custom mechanisms can be written and configured. Our solution makes use of this extension mechanism to connect the Globus Toolkit with our authorization service.

Figure 1 shows the proposed architecture to enable our authorization system for Globus. The architecture provides a security plug-in which integrates within the authorization framework and enables the processing of authorization requests. This means that any authorization request made by a Globus service can be redirected to an external service. In this case, it is sent to our authorization service, named SemanticAuthzService, which is responsible for reasoning about the request and then taking an authorization decision. The service evaluates the request using the security policy and returns whether the given subject can perform the action on the requested Grid service or not. This response is used by Globus to finally enforce the authorization policy by granting or denying the requested access.

![Figure 1: Authorization architecture](image)

It should be noticed that this architecture design keeps the Globus framework and the authorization service decoupled. This enables us to reuse the service in other Grid-related architectures and implementations.

3.1. Linking the authorization service

The integration of the authorization service within the Globus infrastructure is achieved by means of a set of components which link with the Globus authorization framework. This framework consists of an engine that evaluates a chain of configured Policy Decision Points (PDPs for short) to determine if the client making the invocation can access the resource or service. The chain can also be configured with Policy Information Points (PIPs) which can be used to collect attributes about resources, operations, subjects, etc. to be used in the decision making process.

The architecture makes use of a Globus security plug-in which consists of a SemanticAuthzPDP and a set of SemanticAuthzPIP interceptors, as can be seen in Figure 1. The SemanticAuthzPIPs collect attributes associated with authorization requests such as subject, protected resources, action and so on, which are necessary for the authorization decision. Different PIPs can be used for different attributes. For example, there may be a PIP for attributes related with subjects and another one for resource attributes. These attributes are represented in terms of ontology.
instances, which is the expected format for our authorization service. In turn, the SemanticAuthzPDP is really a PDP proxy for Globus. It connects with the SemanticAuthzService, which is the one which really takes the authorization decision, acting as real PDP. The SemanticAuthzPDP provides the information retrieved by the PIPs and asks the service for an authorization decision, which will be returned to Globus for it to grant or deny the access to the requester.

3.2. Managing authorization information

The main component of the architecture is the SemanticAuthzService. This service should handle the multiple ontologies and policies of the Grid participant organizations to make use of that knowledge when taking authorization decisions. Each organization has its own information system describing its users, services, resources and any other relevant domain information. The authorization service needs organizations to provide such information to use it for authorization purposes. This can be achieved by means of the WS-Resource framework (WSRF) [15] specifications, which allow to declare and develop the association between a Web service and one or more stateful resources. WS-Resources are defined as the composition of a Web service and a stateful resource and the WSRF specifications define mechanisms to access, monitor and manage the lifecycle of such WS-Resources. This is used in our solution for organizations to provide and manage their information in their own WS-Resource. The information is represented and kept up-to-date by organizations in OWL ontologies and SWRL rules in order to enable the service to perform the semantic-aware authorization process. Using the WSRF standard technology, information can be kept separate, using a different WS-Resource for each organization.

In this approach where multiple WS-Resources are used, an out-of-band mechanism is needed to create them. The WSRF specifications recommend to follow a factory pattern in which a different service is used to create the WS-Resources and the main service manages their information. Hence, although not depicted in Figure 1 for clarity reasons, there is also a factory service in our proposed architecture in charge of managing the WS-Resource creation.

Handling organization ontologies and policies in different WS-Resources also eases the life cycle management of an organization within the Grid. Figure 2 shows a conceptual sequence diagram to illustrate a WS-Resource lifecycle, which is tied to the organization participation in the Grid.

```
<table>
<thead>
<tr>
<th>Organization</th>
<th>WS-ResourceFactory</th>
<th>SemanticAuthzService</th>
<th>WS-Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>createResource()</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WS-Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>manage()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>destroy()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>initialize()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>manage()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>destroy()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 2: Organization WS-Resource life cycle

When the organization joins the Grid, it uses the factory to create its own WS-Resource. This call returns an endpoint reference which is handled by the organization for future operations on the resource. Then, it makes use of the authentication service to manage its authorization data, accessing it through the appropriate WS-Resource-qualified endpoint reference. Initially, the organization has to define its own ontology and authorization policy, which is stored and handled by the authorization service in the WS-Resource of the organization. Although they are just depicted in the diagram as a single manage() call, different operations (get, add, update, etc.) can be made by the organization in order to manage its knowledge while still being part of the Grid. Finally, if the organization decides to end its participation in the Grid, it sends a request message for the service to destroy its WS-Resource.
3.3. Access request processing

Once the architecture has been set up and organizations have properly created their own WS-Resources, providing their ontologies and authorization rules, a Grid resource access request made by a client will trigger the sequence depicted in Figure 3.

The client request is intercepted by the Globus Toolkit, which has been configured to use the proposed authorization architecture, registering the SemanticAuthzPIPs and the SemanticAuthzPDP. The toolkit uses PIPs to collect information relevant for authorization (e.g. subject and resource attributes). After that, it queries the PDP for an authorization decision, forwarding to it that information. The PDP connects the SemanticAuthzService for it to actually take the decision. The call from the PDP provides request information in OWL, containing the information collected by the PIPs. The SemanticAuthzService uses the TrustManager to get the organizations information to be used for authorization. The TrustManager is able to determine the organizations involved in the request and the trust relationships established among them. It accesses the WS-Resources and, based on the request and the trust relationship information, it returns the information in OWL and SWRL of the corresponding trusted organizations. This information, together with the one provided by the PDP, is passed to the AuthzEngine, which performs the reasoning process.

The AuthzEngine uses an OWL reasoner to infer new knowledge based on the provided information, applying the authorization rules and determining whether they indicate a positive (permit) or negative (deny) authorization. Details about the reasoning process can be found in Section 4. The result of this process is used by the service to generate the authorization response, which is returned to the PDP and to the Globus authorization framework. In case of positive answer, the client accesses the requested resource of the Grid.

4. Authorization policy management

The use of Semantic Web languages like OWL and SWRL provides a generic way of specifying domain descriptions and policies with powerful expressiveness, being able to satisfy the modelling requirements of different domains. However, for the authorization system to understand and enforce the specified policy, the definition of a minimum set of concepts becomes necessary. Some authorization related concepts should be specified in order to provide a positive or negative authorization statement. It should be noticed that the definition of such concepts does not mean a
restriction to the organizations ontologies. They can directly use this set of concepts to express authorization, which are already provided and so they do not need to design a new set. Or they can still define their particular concepts and then provide a few OWL constructors or SWRL rules to map these concepts to those which are needed by the authorization engine to perform the authorization process.

Moreover, a generic and extensible ontology to model the protected resources is provided. Such an ontology defines concepts regarding system resources management, which fits well with Grid environments. Like the aforementioned authorization concepts, this ontology can also be used by organizations which want to use our authorization system without the need to define its own ontology from scratch. This approach has been followed for the proof of concept implementation of this paper exposed in Section 5.

The proposed ontology is based on a well-recognized standard information model, namely the Common Information Model (CIM) [16] created by the Distributed Management Task Force (DMTF). This enables the representation of policy information in a uniform manner and guarantees the necessary extensibility to support any kind of authorization. The main motivations behind the choice of this model are: i) CIM is an standard information model which is independent of the language used to represent it, thus it allows ontological representation in OWL; ii) CIM is a very complete information model covering almost all the different aspects required in information systems, including services, networks, applications, etc., so its OWL representation can be used as generic ontology; iii) there are related standards and technologies grouped under the WBEM [17] specifications which allow to dynamically gather the current state of the system by means of CIM, which may be used by Grid organizations to get information about their systems for authorization purposes; and iv) CIM is free, open source and extensible, ready for new information systems.

Regarding the mapping of CIM to OWL, several representations of CIM in OWL are available such as [9], [18], and [19]. The mapping proposed by [9] holds a high expressiveness in the translation to OWL in contrast to the approach provided on [18], which only preserves RDF expressiveness. Regarding the way in which relationships between CIM elements have been mapped, [19] proposes the mapping of CIM relationships to OWL classes, whereas [9] defends to map CIM relationships to OWL properties. This last approach confer simplicity to CIM models and, in turn, it allows to model semantics applied to these properties. Hence, the approach provided on [9] has been chosen to represent CIM in OWL for our authorization system.

The basic components of an authorization decision (or privilege) are subjects, actions, and targets. A sample authorization decision would be read as follows: it is permitted for subject(s) S to perform action(s) A in the target(s) T. To model the management concepts that are related to an authorization decision, our solution makes use of the set of CIM concepts depicted in Figure 4.

**Figure 4: UML diagram of authorization concepts**

Privilege is the base concept for all types of activities which are granted or denied to a subject for a target. AuthorizedPrivilege is the specific subclass defining current privileges which result of applying the authorization policy rules. The association of subjects to AuthorizedPrivilege is accomplished via the authorizedSubject association. The entities that are protected (i.e. targets) can be similarly defined via the authorizedTarget association.

Subjects are represented by the Identity and Role concepts. The Role concept is used to represent a position or set of responsibilities within an organization, organizational unit or other scope, and it can be filled by people or non-human entities using the memberOfCollection association. An instance of an Identity represents an element that acts as a security principal, i.e., an entity which can be identified and verified by an authentication process. Elements
with identities can be organizational entities, services, systems, etc. The element which is represented by an Identity is described using the assignedIdentity association. Both authorizedSubject and authorizedTarget associations connect with the ManagedElement concept, which is the common superclass (or top of the inheritance tree) for the non-association classes in CIM, thus enabling the definition of any element of the model as subject or target for authorization.

To illustrate these concepts, let us consider an academic organization in which some exams are done with computers. But these exams can only be done in computers belonging to a controlled network whose computers are inside invigilated zones to avoid students cheating the exams. The domain description for this example can be represented using a CIM based ontology and it may be composed by a role that represent the set of students (#Student), the controlled examination network (#ExamNetwork), the examination service (#ExamService), the privilege to access the exam (#AccessExam) and finally the subject (#JohnDoe). Listing 1 shows a fragment of this domain description using RDF/XML [20] syntax, which is the primary exchange format for OWL 2 ontologies:

```xml
<rdf:RDF>
  ...<Network rdf:about="#ExamNetwork">  
  <name>Examination Network</name>
  ...</Network>
<Role rdf:about="#Student">  
  <name>Student</name>
  ...</Role>
<Person rdf:about="#JohnDoe">  
  <name>John</name>
  <surname>Doe</surname>
  ...</Person>
<Identity rdf:about="#JohnId">  
  <instanceID>John Doe</instanceID>
  <memberOfCollection rdf:resource="#Student"/>
  <identityContext rdf:resource="#ExamNetwork"/>
  <assignedIdentity rdf:resource="#JohnDoe"/>
</Identity>
<Service rdf:about="#ExamService">  
  <name>Examination Service</name>
  ...</Service>
<AuthorizedPrivilege rdf:about="#AccessExam">  
  <instanceID>examination:accessExam</instanceID>
  <authorizedTarget rdf:resource="#ExamService"/>
  ...
</AuthorizedPrivilege>
</rdf:RDF>
```

Listing 1: Sample domain description

In this context, the administrator may decide the following authorization policy: if the subject is a student and he is in the controlled examination network, then he is permitted to access to the exam. Rule 1 shows the representation of this policy in SWRL abstract syntax.

The first two lines of Rule 1 define an identity which belongs to the student role. identityContext in line 4 specifies the scope of the Identity, which should be the examination network defined in line 3. In this example the privilege is already defined for accessing the examination service and the rule associates the privilege with subjects to grant them access to the service. This way, privilege instances may be reused for different subjects. Lines 5 and 6 define the privilege which allows accessing the examination service and the rule consequent establishes the authorizedSubject association to grant the privilege to the identity which represents the student.

This rule will be applied to the domain definition in Listing 1 during the reasoning process described in Section 4.2, inferring new knowledge which grants the privilege to access the exam to the student John. Listing 2 shows the OWL representation of the inferred knowledge in RDF/XML syntax.
Identity(?i) ∧ memberOfCollection(?i, ?r)∧
Role(?r) ∧ name(?r, ‘Student’)∧
Network(?n) ∧ name(?n, ‘ExaminationNetwork’)∧
identityContext(?i, ?n)∧
Service(?s) ∧ name(?s, ‘ExaminationService’)∧
AuthorizedPrivilege(?p) ∧ authorizedTarget(?p, ?s)∧ →
authorizedSubject(?p, ?i)

Rule 1: Student in examination network

There are several tools for supporting the definition of domain applications in OWL ontologies. For example, Protégé [21], CO-ODE [22] and SWedt [23] are editing tools to ease the description of ontologies under the Semantic Web languages. These tools successfully deal with OWL and SWRL languages. However, this kind of tools does not offer support for debugging processes when a policy is being inserted into the system. The debugging process during the designing of policies is crucial in order to check the correct behaviour of new policies. Thus, our solution uses the Ontology Rule Editor (ORE) [24], which is a stand-alone application created in our research group and aimed to graphically edit, test, debug and validate ontology rules in any domain. ORE contains a wizard to guide the administrator when editing rules and interpreting the reasoning results. It also incorporates a reasoner which allows debugging the high-level policy definition. It aids the administrator to test a given set of policies before they are applied to the actual system. If some of these policy rules are not behaving as expected, the administrator has the possibility to change the definition before the final policy is enforced to the authentication system.

4.1. Virtual Organization Management

A Grid system is set up among different individuals and/or organizations which share a set of resources in a coordinated manner to reach a common goal, making up what is called a Virtual Organization (VO) [25]. Such a resource sharing should be managed and controlled in order to establish who is allowed to perform what actions.

Usually, each resource provider may want to define its own authorization policy to specify the access to its resources. But also the VO itself may define some common policy which govern the behaviour of the resource sharing. This situation leads to multiple authorization policy specifications provided by different organizations which may also be defined in different terms due to the differences between the organization domains. For example, an academic organization may define their users using the concept student and define its policies based on that concept. On the other hand, a service provider may define its policies based on the concept customer to refer to its users. If both organizations are part of a Grid system, a simple join of their policies may not work, even though the concepts customer and student have the same meaning in both organizations (i.e., an user). This is because of the usage of different syntactic representations which will be considered as different concepts by the system.

Using a semantic-aware approach, each organization (as well as the VO itself) will represent its policies by means of an ontology, making use of its own authorization terms and concepts which are meaningful in its domain. Then, a concept alignment can be established in order to relate concepts between different domain ontologies. This alignment can be done using the constructors provided by OWL and SWRL rules. Taking up the previous example again, the
concepts \textit{customer} and \textit{student} may be aligned by means of the \textit{equivalentClass} construct provided by OWL - i.e. specifying \texttt{equivalentClass(student, customer)}.

Nevertheless, not all the organizations of the Grid should know about the rest and provide a mapping for all domains. The VO is established in order to solve a need by sharing resources and it may define its own ontology to represent the particular concepts which are significant in its domain, i.e. relevant for a successful resource sharing among the different organizations. This ontology is agreed between the participating organizations of the Grid and can be used as reference point for concept alignment. Any organization which wants to join and share resources provides the set of OWL constructs and SWRL rules to relate the concepts in its domain with the ones in the VO domain. Figure 5 depicts this approach.

Using this approach, the VO may, for example, define some policies for users being of legal age. It can make use of ontology concepts such as \textit{User} and \textit{UserBeingOfLegalAge}. The concepts \textit{Customer} and \textit{Student} may be defined as \textit{User} by specifying \texttt{equivalentClass(Student, User)} and \texttt{equivalentClass(Customer, User)}. Then, using SWRL rules, the unified concept \textit{User} may be used to define an user being of legal age as any user which is aged above 18. The rule may be something like the one specified in Rule 2.

\[ \text{User(?u)} \land \text{age(?u, ?a)} \land \text{swrl:greaterThanOrEqual(?a, 18)} \]  
\[ \rightarrow \]  
\[ \text{UserBeingOfLegalAge(?u)} \]

Rule 2: User being of legal age

Thus, policies which apply to users being of legal age will apply to both students and customers aged above 18, while neither the service provider nor the academic organization should have knowledge of each other. They only have to know about the \textit{User} concept defined by the VO.

For this concept alignment to be achieved, a joint of the information provided by organizations becomes necessary. In our authorization architecture, information of different organizations is kept separate for privacy purposes by means of the WSRF specifications. However, each participating organization is also able to specify a set of other organizations which are trusted by it and to which its information can be safely shared. This serves to model a trust relationship...
among organizations, meaning that an organization of the Grid trusts on the knowledge provided by another one. This relationship is used by the TrustManager upon a request to only use the knowledge of those organizations which trust on each other when taking the authorization decision. Trust on the VO is assumed since the VO arises from an agreement between the participating organizations to form the Grid and concept alignment with the VO should be performed to take its authorization policy into account. The rest of organizations are considered untrusted unless explicitly defined by this relationship. Moreover, for the knowledge of two organizations A and B to be joint, they should trust on each other, i.e., A should specify its trust on B and B should specify its trust on A.

The TrustManager is in charge of providing the information to be used during reasoning for a given authorization request. The information managed by the reasoner is kept in what is called a knowledge base (or KB for short), which is a data structure in which both the original ontology and the new inferred information is stored. A different KB is maintained for each organization, containing its own knowledge together with the knowledge of its trusted organizations. The TrustManager keeps these KBs up to date by updating them when a trusted organization changes its domain information. Then, since each resource provider defines the access to its resources, the reasoning process is performed using the KB of the organization to which the request target belongs.

It should be noticed that at this time the trust relationship is not transitive, i.e., if organization A trusts on B and B trusts on C, it does not imply that A trusts on C. Management of transitivity may become a complex task and it is still not supported. Although our authorization architecture has been designed with trust management support and a first approach is provided, the development of a complete solution including trust management is a future work in our research group and out of the main goal of this paper.

4.2. The reasoning process

Authorization decisions are taken during a reasoning process which is performed by an OWL and SWRL reasoner in the SemanticAuthzService. The main operations for this reasoner are:

- Inference. New knowledge about the domain is inferred using the information which is available in the ontology. Inference about SWRL rules which represent the authorization policy derives new individuals or OWL instances which are used to make the authorization decision. An example of inference can be found in Section 4, when the policy rule shown in Rule 1 is applied to the domain defined in Listing 1. The inferred knowledge is shown in Listing 2.

- Validation. The OWL language allows constraints to be expressed; the validation operation is used to detect when such constraints are violated by some data set. In other words, the validation consists in a global check across the schema and instance data looking for inconsistencies.

- Querying the ontology, including instance recognition and inheritance recognition. The former consists in testing if an individual is an instance of a class expression and the latter in testing if a class is a subclass of another class or if a property is a sub-property of another. This enables the formulation of generic queries referring to abstract concepts, being the system able to recognise instances belonging to concrete subclasses or sub-properties of such a concept. For instance, let us suppose that role hierarchy is supported in the domain defined in Listing 1 and that a role is defined as superclass for all members of the University (#UniversityMember). This role may have generic permissions to access basic services like access to the University network. Thus, if a query is defined to look for who is allowed to access the network, John Doe will be returned, since it belongs to the #Student role (instance recognition) which, at the same time, inherits from #UniversityMember (inheritance recognition).

Figure 6 depicts the elements which take part during the reasoning process. The domain knowledge and constraints are specified in form of OWL ontology. Different ontologies can be used and constructs and rules for concept alignment can be specified as exposed in Section 4.1. Two different kinds of rules can be also distinguished: policies and meta-policies. The former represent the authorization policies defined by the administrators, like the one shown in Rule 1. These policies guide the inference during the reasoning process to make a positive or negative authorization decision. Meta-policies are policies which provide additional information for authorization. They are rules which can be used to define advanced semantics of the protected resources which need to go beyond what can be specified by means of OWL constructs, as the one specified in Rule 2 of Section 4.1. This is an added value which provides an
extensible mechanism for describing the semantics of the domain model, as well as enhanced constraints which can be used for policy conflict detection and resolution. Details about the usage of meta-policies for conflict detection, including an example, can be found in Section 4.3.

During reasoning, the reasoner uses the knowledge base which contains the OWL ontology together with the policies and meta policies defined in form of SWRL rules. Two kinds of reasoning can be identified: OWL reasoning and SWRL reasoning. The former implies the aforementioned operations of querying, validation and inference about the ontology, while the latter consists in performing inference with SWRL rules. The inference process (about both the OWL ontology and the SWRL rules) generates new knowledge which is used to update the KB. Validation is also performed, checking the ontology for inconsistencies or constraint violations. Finally, when the AuthzEngine queries the reasoner for an authorization decision, this is taken based on the presence of certain instances and associations on the KB. Those instances represent an authorization definition according to our CIM based model depicted in Figure 4 and they may be present due to statical definitions provided by the administrator or they may be generated by inference during reasoning.

Listing 3 shows the authorization query in abstract syntax which gets the privilege allowing a subject to access to a given target. Namely, it explores the KB looking for an AuthorizedPrivilege instance which is associated with the given subject and target. The query has two parameters: the subject or identity which is trying to access; and the target representing the Grid resource which is being accessed. If the query does not return any privilege instance, the AuthzEngine denies the subject accessing the target (deny by default).

For this reasoning process to be efficient, an incremental approach is used. Some OWL reasoners like Pellet [26] support incremental consistency. This functionality enables the reasoner to just take into account the new knowledge which may have been added since the last reasoning performed with the same KB. In our approach, a KB is maintained for each organization with its knowledge and the one of its trusted organizations. This KB is updated on every request and the new knowledge inferred during reasoning is also kept in it. Since this KB remains the same for different requests, incremental consistency can be used to make the reasoning process faster. If the organization domain and
the authorization policy is not changed, the new knowledge between reasoning processes is in many cases only the information provided by the PIPs for the current request. For initial requests, this information may be unknown for the system, but when the authorization system has been up for some time, the KB will probably contain information about most of the system and few new knowledge will really be included as new for reasoning. Moreover, the reasoner is also able to detect knowledge which is already present in the KB. Thus, if the information provided by a PIP is already in the KB (probably provided by another PIP from a previous request), it will not be validated again, since it was validated in a previous consistency checking process.

4.3. Policy conflict analysis

An important functionality of the reasoner is that it enables the detection and resolution of conflicts [27]. Conflicts raised by evaluating policy rules may be categorized into two main types, depending on whether the conflict depends on the application domain or not. In the first case, the conflicts obtained are called semantic conflicts, because they are generated by using information related to the current state of the system. They are really difficult to detect, and their appearance is conditioned to the dynamic state of the application domain. In the second case, we have syntactic conflicts as they can be detected by simply looking at the rules structure. This second type of conflict occurs irrespective of the state of the particular application domain and may be the result of specification errors in the policy or they may also be legitimately derived from other rules. Moreover, the heterogeneous environment of a Grid composed by different organizations may lead to the rising of policy conflicts. Conflicts may appear between different organization policies or even between the policies of an organization and the policies of the VO.

Focusing on authorization policies, modality conflicts are syntactic conflicts that can be detected from the rule syntax that defines these policies. A modality conflict occurs when two or more policies with modalities of opposite sign refer to the same subjects, activities and targets. An example of this type of conflict is observed when two different policies about a subject (e.g. John) forbid and permit to make the exam at the same time. Thanks to the automatic generation performed by the reasoner, the system is able to identify a policy conflict when two disjoint properties appear simultaneously (i.e. inconsistency). Several techniques may be applied to make this inconsistency for conflict detection. As a first approach, our proposal defines an unauthorizedSubject association which is declared as disjoint with authorizedSubject, using the disjointDataProperties construct provided by OWL. The inconsistency appears when, due to some policy, the reasoning process infers an instance unauthorizedSubject between John and the privilege to access the exam (meaning that John is not authorized to access the exam) and, at the same time, due to some other conflicting policy, the rule reasoner also infers a new instance of authorizedSubject between them (meaning that John is authorized to access the exam). The inconsistency is detected by the reasoner during validation because an individual cannot belong to two disjoint properties simultaneously. Listing 4 shows the inconsistency in OWL using RDF/XML syntax.

```
<AuthorizedPrivilege rdf:about="#AccessExam">
  <instanceID>examination:accessExam</instanceID>
  <authorizedTarget rdf:resource="#ExamService"/>
  <authorizedSubject rdf:resource="#JohnId"/>
  <unauthorizedSubject rdf:resource="#JohnId"/>
...</AuthorizedPrivilege>
```

Listing 4: Privilege with inconsistency

On the other hand, semantic conflicts are domain dependent and their detection is based on some meta policies describing undesirable situations which may appear in the domain, in order to detect them and avoid their occurrence. An example is observed when two different policies about a subject permit him to access the exam at the same time they allow him to access the exam responses. This conflict is semantic because the subject (e.g. John) may get the responses to the exam before sending his own responses. A meta policy may be defined to detect such situation, forbidding the subject to get the exam responses. Rule 3 shows the representation of this meta policy.

Once the conflict has been detected, the rule consequent states that the subject is not authorized to get the exam responses by establishing the unauthorizedSubject association between the privilege and the subject. This property will cause an inconsistency in the knowledge base with the already existent authorizedSubject property, since they are defined as disjoint properties.
\[ \text{Identity}(?) \land \\
\text{AuthorizedPrivilege}(\text{#AccessExam}) \land \text{authorizedSubject}(\text{#AccessExam}, ?i) \land \\
\text{AuthorizedPrivilege}(\text{#AccessResponses}) \land \text{authorizedSubject}(\text{#AccessResponses}, ?i) \land \\
\rightarrow \\
\text{unauthorizedSubject}(\text{#AccessResponses}, ?i) \]  

Rule 3: Simultaneous access to exam and responses

Finally, conflict resolution is necessary to provide an automatic solution for detected conflicts. Prioritization of policies is a suitable technique for resolving the authorization conflicts [27][28]. Conflicts may appear between rules defined by different organizations. In this case, assigning a priority to each organization may resolve the conflict. For instance, the VO may be defined with a higher priority and, thus, its policy should apply in case of conflict. Conflicts between rules of the same organization should also be resolved prioritizing the rules or even prioritizing the authorization decision which should be taken. For instance, specifying that negative policies have priority over positive ones (deny by default). Once detected, a decision can be made to resolve the conflict, taking the decision with higher priority.

It should be noticed that this section has introduced how both semantic and syntactic conflict detection and resolution are supported by the proposed authorization approach. However, conflict detection and resolution are quite complex tasks, including conflict analysis and source of conflict discovery. These are ongoing research lines in our research group and a complete discussion about these topics is out of the scope of this paper.

5. Implementation and statistics

This section provides some details about the proof of concept implementation which has been developed to evaluate feasibility and performance of our authorization solution. Some statistical results achieved are also exposed herein. Thus, as will be shown later, these statistics deal with the evaluation of the average authorization response time when a user tries to access to a resource in the Grid, as well as with the evaluation of the reasoner behavior in charge of taking such a decision.

With regards to the implementation used to perform the testbed, it is worth taking into account some considerations. Firstly, the SemanticAuthzService and the WS-ResourceFactory described in Section 3.1 have been developed as Globus Web services making use of Introduce [29] as a tool for aiding on Web services development.

Moreover, in order to abstract the SemanticAuthzService from the KB management, a Java library has been developed. This library defines a simple interface which allows the service to manage the organization ontologies, as well as to perform the authorization query in charge of making the authorization decision. The query is executed using SPARQL [30], the query language fostered under the Semantic Web technologies.

Currently, there are several suitable implementations of DL reasoners such as Pellet [26], Jena [31], KAON [32] and the one proposed by FaCt++ [33], among others. Our Java implementation makes use of Jena as a general Java API to manage ontologies and Pellet as DL reasoner. Namely, it uses Jena version 2.6.2 and Pellet in its version 2.0.0. The main reason by which Pellet has been chosen as reasoner is due to the fact that it supports high expressiveness dealing with OWL 2 ontologies and it is also able to perform incremental consistency checking. Likewise, Jena is nowadays the standard de facto Java library to manage ontologies.

On the other hand, it is worth noting that the statistics have been achieved making use of virtualization technology. Thus, an Intel Xeon machine with one processor at 2GHz split into 4 Cores holding 4 Gb of RAM has been used as host to install a VMware Server, in its version 2.0.2. The VMware virtual machine where has been install the Globus Toolkit 4.0 to execute the testbed has been configured with 3 Gb of RAM, using Ubuntu 9.10 as Operating System. Besides, the Globus Toolkit Java WS Core container has been customized to 2.5 Gb of RAM to hold the authorization solution.
5.1. Reasoner performance

An important part of our architecture which widely affects the performance is the reasoner in charge of making the authorization decision which is taken based on ontology instances present on the KB. In order to check the performance and behavior of such a reasoner when it is under different work loads, several tests have been done using different amount of individuals, i.e. OWL instances, present in the KB.

A way of measuring the performance of our authorization solution is making different complexity executions. These executions are performed using different sets of work loads in the KB which make the test to become more and more complex. Thereby, the complexity is achieved by introducing new knowledge in the ontology. An ontology can be divided into TBox (terminological) and ABox (assertional) components. The former contains the schemas and vocabularies that define domain concepts, whereas the latter is populated with instances of these concepts. Thus, the complexity of executions is incremented by increasing the number of individuals present in the ABox component of the ontology what, in turn, increases the number of statements hold in the knowledge base. In order to add new instances to the KB, the set of operations provided by the SemanticAuthzService to manage the ontology are used as exposed in 3.2.

The number of different kinds of individuals contained in the ABox for a specific execution is referred as population. Thus, a population represents the knowledge provided by an organization at a given moment. The addition of new instances leads the KB to reach the next population state, making it necessary to perform the KB consistency checking again. It should be noticed that the number of axioms or statements that are hold in the ABox differs from the number of individuals which are present in the ontology, since more than one axiom is usually necessary to represent one individual.

As stated in section 4, the CIM to OWL representation proposed by [9] has been used as ontology to model the domain concepts in this implementation. Each population is composed by individuals which are represented by instances of different OWL classes defined in the authorization model for Grid described in Figure 4, i.e., CIM_Identity, CIM_Role, CIM_AuthorizedPrivilege and CIM_ManagedElement. Figure 7 depicts the percentages of every kind of individual which have been used to make up each population of the testbed. These percentages has been selected in order to represent a realistic authorization system in a Grid. Thus, the individuals for a given population have been randomly generated, but in a driven way in order to achieve the desired distribution for the testbed to better fit a real scenario.

Another issue to be considered is the way the populations sizes are established since this size will be used to evaluate the reasoner scalability. In this sense, it is very common to make use of an exponential function to establish the size of these populations. It should be notice that the individuals percentage distribution described above is maintained for each population regardless its size.

The testbed makes use of 10 incremental populations. The following function, which determines these ten populations size, has been empirically achieved, starting on the basis that, for our testbed, the biggest population must be the one which shows unrealistic performance results. Thus, the biggest population in this testbed has about 27000
individuals, since for this amount the reasoning process needs more than 10 seconds to perform the reasoning, which is usually considered as an unacceptable time for any authorization system.

\[ f(x) = 300 \cdot e^{(x/2)} \]

The variable \( x \) takes integer values from the interval \([0, 10]\). Table 1 shows the relationship between the number of individuals for each population used in the testbed and the corresponding number of statements representing such individuals.

<table>
<thead>
<tr>
<th>Population</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals</td>
<td>300</td>
<td>500</td>
<td>900</td>
<td>1400</td>
<td>2300</td>
<td>3700</td>
<td>6100</td>
<td>10000</td>
<td>16400</td>
<td>27100</td>
</tr>
<tr>
<td>Statements</td>
<td>1500</td>
<td>2800</td>
<td>4700</td>
<td>8300</td>
<td>14100</td>
<td>25200</td>
<td>46200</td>
<td>86200</td>
<td>167500</td>
<td>336000</td>
</tr>
</tbody>
</table>

Table 1: Number of individuals and statements by population

As stated before, these populations are generated with individual distributions trying to represent realistic scenarios. For instance, the population number six which has 3700 individuals, could represent a scenario where a Institute of Technology with 1500 students and 350 professors have rights to access to 555 different managed elements, such as, PCs, rooms, resources, etc.

These populations will be used to quantify the time that our authorization system takes to check the Knowledge Base consistency and perform the authorization query on the KB. It should be noticed that the reasoning time is determined not only by the number of statements hold in the ontology, but also by the TBox complexity itself. However, the schema and vocabularies that define the domain concepts, making up the TBox, are the ones present in [9] ontology and they remain the same for all the executions. Thus, this complexity is not explicitly shown in statistics since the Tbox is not changed and this time component is constant.

As exposed in Section 4.2, for each authorization request processing, the reasoner performs the validation and query operations on the KB. Figure 8 depicts the Pellet reasoner performance for both the time it spends making the validation or consistency checking process and the time spent carrying out the SPARQL authorization query once the KB is valid. This query searches instances in the KB which may indicate a positive (permit) or negative (deny) authorization for the requested Grid resource, as stated in section 4.2. Moreover, the statements hold in the KB are added incrementally, i.e., populations grow by adding new individuals, keeping the ones present in the previous population what emulates an organization in the grid system adding new information to its existing domain model.

As shown in figure 8, the greater number of statements the KB holds the more time it takes for the reasoner to perform the consistency checking task. However, except for the first query, the time the reasoner needs to query the KB
grows quite slower compared with the time spent in the consistency checking. The first query takes more time since
the reasoner has to perform the Classification reasoning process which computes the subClassOf and equivalentClass
inferences between all ontology named classes and properties. This time appears just in the first execution, since the
TBox is not changed as stated before and it remains the same for the rest of queries.

5.2. Authorization request testbed description

Different simulations have been carried out with the aim of evaluating the average time needed to get an authoriza-
tion response from our authorization system in Globus Toolkit. The test is performed from the moment when a
user tries to access to a resource shared in the Grid, until the authorization system gives the authorization response. It
makes one or more requests evaluating the KB when it is under incremental work loads. The overall time has been
split in the test in four different times in order to be analyzed and understood properly. Figure 9 depicts these four
times and the main tasks performed during such periods.

The first time, referred as Globus Toolkit Time, deals with the period spent by the Globus Toolkit to process a
user request and redirect it to our SemanticAuthzPDP. A user registered in the Grid, with his own certificate, tries
to access to a dummy Grid Web service called HelloWorldService. In the testbed, the user has been previously
authorized to access to this Web service in the authorization system, and therefore, the ontology individuals which
grant the user to access the Web service are included in the KB.

This time includes the elapsed time introduced by the Toolkit for authenticating and securing the HTTP commu-
nication. Thereby, our solution make use of message level security by means of the Globus GSI Secure Conversation
mechanism [34], where the client establishes a context with the server before sending any data. Once the context es-
tablishment is completed, the client can securely invoke an operation on the service by signing or encrypting outgoing
messages using the shared secret captured in the context.

When the HTTP message arrives from the client, the Globus SOAP engine invokes several security related handlers
in what it is called handler chain. Thus, the WS-Security handler, searches the message for any WS-Security headers
extracting from them any keying material. Then it populates a peer subject object with principals and any associated
keying material. Next the Security policy handler, checks that incoming messages fulfill any security requirements
the service may have.

Afterwards, the Authorization handler verifies that the principal established by the WS-Security handler is au-
thorized to invoke the service. At this point, since the HelloWorldService has been previously configured in the
security descriptor to make use of our SemanticAuthzPDP, the Globus authorization request reaches our PDP imple-
mentation. It is important to remark that the Globus Toolkit Time presented in this testbed is out of our authorization
system implementation. However, it is taken in order to evaluate the whole time required to obtain an authorization
request response in the Globus Toolkit when making use of our solution.

The PDP Time depicted in Figure 9 deals with the period spent by our SemanticAuthzPDP. This component
implements the Globus Toolkit org.globus.security.authorization.PDP interface and behaves like a gateway
that redirects the Globus authorization request, through an HTTP call, to the SemanticAuthzService. This con-
nection is also secured with message level security by means of the GSI Secure Conversation mechanism explained
above. It is also worth mentioning that, in this testbed, the PDP Time does not include the time spent by third orga-
nizations PIP implementations needed to obtain attributes related to the user request. This time does not depends on
our authorization implementation since each organization should provide its own PIPs and therefore the time is quite implementation specific.

The elapsed time introduced by the SemanticAuthzService can be mainly split into two periods. On one hand, the time spent by the reasoner to check the consistency of the KB, shown as Consistency Checking Time in Figure 9. On the other hand, the time spent carrying out the SPARQL authorization query, depicted as Query Time in the figure.

Listing 5 shows the SPARQL query that has been used in our proof of concept implementation which is a realization of the abstract one exposed on Section 4.2. As can be seen, the query is self-configurable and looks for any CIMAuthorizedPrivilege individual in the KB which allows a subject to access to a given target. In this case, the target is a Web service modeled by the CIM class CIM_Service. Moreover, since Globus Toolkit identifies the subject by means of certificates, and therefore, such certificates information is available in our implementation, the SPARQL query identifies the subject with the CIM property CIMPublicKeyCertificate.Subject. The CIMPublicKeyCertificate is, at the same time, assigned to the CIMIdentity which represents this subject in the system.

Listing 5: SPARQL authorization query

```
PREFIX cim: <http://gom.kwgrid.net/ontology/cim/2006/10/02/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?privilege
WHERE {
  ?privilege rdf:type cim:CIM_AuthorizedPrivilege .
  ?privilege cim:associationCIM_AuthorizedSubject ?identity .
  ?privilege cim:associationCIM_AuthorizedTarget ?service .
}
```

In order to set up the SPARQL authorization query, the implementation makes use of the default Globus PIP X509BootstrapPIP. It collects attributes from the X509 user certificate and the Globus Toolkit makes them available for the SemanticAuthzPDP and, in turn, for the SemanticAuthzService which actually replaces the values in the query and executes it. Therefore, the variable subject in the query is replaced by the Distinguished Name (DN), e.g. ”/O=Grid/OU=GlobusTestbed/OU=simpleCA-john-desktop/CN=John”. Likewise, the variable target is replaced in the testbed by “http://org.umu.security.grid.semantic/HelloWorldService” since it is the name given to the HelloWorldService which acts as target in our testbed.

5.3. Authorization request performance

It is important to remark that the reasoner always performs the consistency checking for each authorization decision request, but if the KB has not been modified this process does not take any time because the KB is already reasoned and consistent. Therefore, the average authorization time response can be relatively lower in systems characterized by few changes in the KB, where the consistency checking operation takes no time for many consecutive authorization requests. Such static systems could be those which are in production during a long time and therefore the system is relatively steady. On the other hand, an example of a system under frequent changes could be the one where new organizations joins the Grid or leave it very often.

Figure 10 depicts four different stats representing four different scenarios that could take place in a Grid, arranged from dynamical to more statical systems. It permits to show the behavior of our authorization solution when it is working in different scenarios.

The first graph represents a dynamic system where the KB is changing continuously. In such systems, the reasoner is forced to spend time checking the KB consistency in every authorization request. As it can be seen in the graph, the Consistency Checking Time gets a high percentage of the overall time required by a request. Initially, when the request is executed for the first time, the Globus Time and the PDP Time gain prominence of the overall request time. This is due to the fact that the first time the user makes the HTTP call to the HelloWorldService it is necessary to establish the GSI Secure Conversation context. Therefore, it has to configure the security communication
context between the user and the Globus Toolkit and also the communication between the SemanticAuthzPDP and the SemanticAuthzService.

The second graph denotes a kind of system where 10 authorization requests occur before the KB changes. The results showed in this graph represent the average times for these 10 requests. During 9 of these 10 requests the Consistency checking Time is zero, therefore, the average Consistency checking Time for these 10 executions is very low. This leads to the reduction of the overall average authorization request time compared with the system situation depicted in the first graph. Moreover, this implies that the Query Ontology Time and the PDP Time represent a greater percentage of the total request time regarding to the previous system situation. In addition, as expected, the time required to perform the SPARQL query is greater when the number of statements hold in the KB grows up.

The third graph corresponds with a common scenario where up to 100 authorization requests can be performed before the KB changes. An scenario like this could be the aforementioned one regarding the Institute of Technology, where there are a great amount of students and their rights to access to the resources hardly ever change, making the KB to be fairly steady. A general deduction looking at this graph is that the average Consistency checking Time is decreased when the system situation become more static. Moreover, it is also important to bear in mind that, although it is not showed in the graphs, the first SPARQL query done just after the KB changes spends more time than the following ones as explained previously. Additionally, with the exception of the first population, the Globus Time and the PDP Time are quite stable during all the executions for every population.

The forth graph symbolizes a more static scenario where the KB hardly ever changes. Thus, up to 1000 authorization requests can be done without the need of perform reasoning on the KB. The average time spent checking the consistency when the system is in such a situation is almost irrelevant, comparing with the average time spent on the Query Time. Thus, as it can be seen in the graph, the percentage of the Query Time is the greatest time in such a system. This last fourth graph shows that the overall average time in a static system like this, is very low compared with the dynamic one shown in the first graph.

To sum up, this section has shown that the reasoning and query times required by our solution to take the autho-
rization decision is quite acceptable when high populations are hold in the KB. Furthermore, the statistics prove that the time required by our solution to give an authorized response compared with the overall time the Globus Toolkit requires to process the authorization request is also quite acceptable. This statement becomes especially evident in a static scenario where the KB is stable and our authorization approach reduces the time required to reasoning over the KB, and therefore, to make the authorization decision.

6. Discussion

Several active working groups and projects are devoting their efforts to address some of the issues related to authorization in Grid. Shibboleth [1] is a service which implements widely used federated identity standards, mainly the Security Assertion Markup Language (SAML) [14] and it is able to provide cross organization access to Web resources. Its interoperability with the Globus Toolkit authorization framework is provided by the GridShib project [35]. It allows Globus to query a Shibboleth attribute authority to obtain attributes regarding the requester. Authorization decisions are then made based on these attributes. Another approach is the Virtual Organization Membership Service (VOMS) [2]. VOMS is a system for managing authorization data within multi-institutional collaborations. In this solution, user attribute assertions are issued in the form of X.509 attribute certificates. Then, some PIP and PDP allow Globus to access and process VOMS attribute certificates and take an authorization decision based on them. These approaches are based on standardized attribute sets which may not fit the organization requirements to define the domain concepts needed by policy definitions. Although attribute sets are usually extensible by adding new types of attributes, these solutions do not allow the definition of semantics related to the elements they are protecting and administrators may find it difficult or they may even be unable to express the desired authorization policy based on these representations. In our approach, a model representation is used, providing more expressiveness and being able to better fit the organization domain representation. Moreover, Semantic Web technologies and languages allows the addition of semantics to these models, enabling them to provide a representation closer to the real world.

A commonly used solution to provide policy based authorization in Grid systems is PERMIS [3][4]. PERMIS is a privilege management infrastructure that uses the principles of Role-Based Access Control (RBAC) to manage privileges and policies to make authorisation decisions. The PERMIS authorization system can be combined with Shibboleth to provide a more complete authorization system for Globus [36]. This integration uses the authorization framework in Globus to collect attributes from Shibboleth and passes those attributes to PERMIS through a custom PDP to render an authorization decision. However, PERMIS policy definition language lacks of semantic expressiveness and sometimes this may lead to a need for the administrator to previously refine the policy he has in mind in order to express it with the constructions provided by the language. Although administrators are usually able to do such a refinement, this may result in a complex policy definition with lots of rules and it can become a difficult task or they may even be unable to achieve an adequate policy definition. In this sense, our approach allows the administrator to combine the definition of semantics in the domain model together with the expressiveness of SWRL rules to specify policies using definitions which are closer to a natural representation of what the administrator has in mind and thus reducing the difficulty and the risk of mistakes when defining the policy.

A policy based solution which allows semantic expressiveness is provided by KAoS [37][38]. KAoS is a set of platform-independent services, allowing fine-grained policy-based management of Grid services on the Globus platform. By providing an interface between Globus and KAoS, it enables the use of Semantic Web to manage GSI-enabled Grid services. However, KAoS lacks of complete solution with reasoning capabilities that facilitates the policy conflict detection and resolution. Some alternative approaches to represent and reason about policies for distributed systems include Rei [39] and Ponder [40]. Although Rei employs a powerful logic-based policy language, it lacks of any support in policy enforcement and requires extra effort from developers to integrate it into existing platforms such as the Globus Toolkit. On the other hand, the policy actions of Ponder can be directly implemented in Java with little additional effort, and has resulted in use for practical applications. However, Ponder inability to change policies at runtime and its low level of abstraction of policies reduces its capacity to meet the requirements of a dynamic environment.
7. Conclusion and future work

The management of authorization for the resources and services offered across a Virtual Organization may result in a complex task which should be faced by Grid administrators. Policy based approaches for authorization provide an intuitive and scalable way for administrators to keep large information systems under control. Policies in these approaches are specified by means of rules which are based on domain concepts which should be also modelled by the authorization solution. Policy and domain languages with high expressiveness, which allow the representation of the resources to be protected and their semantics, would ease the authorization management task.

This paper has presented a semantic-aware approach enabling the dynamic management of authorization by combining Semantic Web technologies with Grid authorization systems. High expressiveness for policy and domain definition is provided, reducing the gap between abstractions and reality to aid administrators in authorization management. Ontologies are used to provide a representation of the underlying information system and the resources to be protected. This allows the inclusion of semantics, providing administrators with language constructors to define object, property and cardinality restrictions, among other advanced features for domain definition. The usage of Semantic Web technologies and languages endow the authorization system with automated reasoning capabilities, making the system able to infer additional information not explicitly specified and to perform a formal validation of the constraints specified in the ontology. These reasoning capabilities enable advanced features like concept alignment and policy conflict detection and resolution.

Multiple authorization domains are supported by the approach exposed in this paper, dealing with heterogeneity of Grid systems and enabling organizations to use their own domain concepts without having knowledge about the rest of participant organizations. An automatic concept alignment is performed to use all this heterogeneous information when taking authorization decisions. The proposal also represents one step towards the automatic management of security services, considering not only authorization services, but also providing additional reasoning mechanisms to cope with issues such as detection and resolution of conflicts between different Grid management rules.

Finally, some performance and statistics have been provided to show the feasibility of the solution. A proof of concept implementation has been tested in Globus against several system scenarios, from more dynamical to more statical ones. Different work loads have been employed in the testbed, evaluating the solution by using populations with an increasing number of individuals present in the knowledge base. The provided statistics show the behaviour of the reasoning and query processes under these different situations, achieving acceptable authorization times.

As statement of direction, we are currently working in advanced conflict detection and resolution techniques that can provide a semi-automated answer when a conflict appears. The first effort is being focused on syntactic conflicts, where different techniques from highly distributed scenarios, such as agent models and computation, are being analysed. We are also extending the current authorization architecture to cover some additional requirements, such as advanced privacy and trust management for organization knowledge, as well as testing the architecture with other Grid-based applications.

References
