Using Electronic Institutions to secure Grid environments

Ronald Ashri[1] and Terry Payne[1] and Michael Luck[1] and Mike Surridge[2] and Carles Sierra[3] and Juan Antonio Rodriguez Aguilar[3] and Pablo Noriega[3]

[1] University of Southampton, UK
{ra,trp,mml}@ecs.soton.ac.uk
[2] IT Innovation, Southampton, UK
ms@it-innovation.soton.ac.uk
[3] IIIA-CSIC, Spain
{sierra,jar,pablo}@iiia.csic.es

Abstract. As the technical infrastructure to support Grid environments matures, attention must be focused on integrating such technical infrastructure with technologies to support more dynamic access to services, and ensuring that such access is appropriately monitored and secured. Such capabilities will be key in providing a safe environment that allow the creation of virtual organisations at runtime. This paper addresses this issue by analysing how work from within the field of Electronic Institutions (EIs) can be employed to provide security support for Grid environments, and introduces the notion of a Semantic Firewall (SFW) responsible for mediating interactions with protected services given a set of access policies. An overarching guideline is that such integration should be pragmatic, taking into account the real-life lessons learned whilst developing, deploying and using the GRIA infrastructure for Grid environments.

1 Introduction

The Grid Computing paradigm [8] is aimed at supporting access to a variety of computing and data resources across geographical and organisational boundaries, to enable users to achieve (typically) complex and computationally intensive tasks. More specifically, the "Grid Problem" has been articulated as providing the means to support virtual organisations that can draw together different capabilities from across the Grid domain, to deliver services that might not otherwise be possible [6]. In attempting to realise this vision, research and development over recent years has focussed on directing Grid environments towards establishing the fundamentals of the *technical infrastructure* required, as represented by infrastructure development efforts such as the Globus toolkit [9], and standardisation efforts such as OGSA [16] and WS-Resource [2].

However, while such technical infrastructure is necessary in providing an effective platform to support robust and secure communication, this largely omits consideration of the other *higher-level* issues that need to be addressed before we can achieve the goal of formation and operation of virtual organisations at run-time based on a dynamic selection of services [8]. In particular, whilst low-level security concerns (including encryption, authentication, etc) are addressed, the problems of describing authorised

processes and the policies that are associated with those processes is largely ignored at this level. The requirement here is to specify which services are allowed to participate in the virtual organisation and what they are permitted to do.

If we consider virtual organisations in the context of agent-based computing, we can regard this problem as analogous to that of defining an *Electronic Institution (EI)*. Electronic Institutions, as defined in [3], can provide the necessary conceptual framework for describing the allowed participants in a virtual organisation as well as the permitted interactions in any given state. As such, they have proven useful in providing structured regulatory environments for heterogeneous external agents or users (in a broader sense). Furthermore, they are supported by tools such as ISLANDER [4], which can facilitated the process of defining an institution.

In this paper, we present a way of making use of such technologies in response to a specific set of needs for Grid applications, identified following practical experience gained through the development of the *GRIA (Grid Resources for Industrial Applications*) infrastructure [14]. Unlike Globus, GRIA was designed to support business interactions, and although it does not currently make explicit use of agent technologies, some of its underlying concepts resonate well with an agent approach. As such, it provides an ideal and flexible framework that could exploit agent technology to provide effective solutions for some of its current limitations.

In particular, we describe how EIs can be applied within the context of a Grid security device, and introduce the notion of a *Semantic Firewall*. The purpose of the Semantic Firewall is to protect Grid services by monitoring all external interactions with those services. It's key functionality is to ensure that all interactions with protected services fulfil the following criteria:

- The encountered interactions are those *expected*, given the agreed aims of the interaction and the current state of execution of a defined interaction protocol [1];
- The interactions must satisfy any security requirements associated with the interaction protocol.

The application of EIs for describing and subsequently monitoring interactions within the context of a Grid application represents one of the primary efforts in demonstrating (in practical terms) how agent technologies can be used in Grid environments. Whilst the perceived benefit of doing so has already been argued by Foster et al [7], this work represents a tangible example that realises this vision. In addition, it also demonstrates how such technologies can be *pragmatically* applied without requiring a drastic reconfiguration of existing Grid infrastructure, or the way in which Grid-developers design services. This is a significant issue since uptake of agent technologies is notoriously hard to achieve in new environments [17]. Thus, the ability to introduce agent-based principles without a significant shift in the status quo, whilst adding value within a Grid Infrastructure is a key contribution.

The paper is structured as follows. In the next section we briefly describe the GRIA infrastructure and provide an example of its operation (section 3) that we use throughout the paper. Subsequently, in sections 4 and 5 we introduce the notion of the Semantic Firewall, and briefly describe Electronic Institutions. We then discuss in section 6 how we map GRIA concepts on to EI concepts and provide a concrete example of that mapping (section7). The paper concludes in section 8.

2 The GRIA Framework

The GRIA framework is a Grid infrastructure developed using just the basic web service specifications, as part of the EC IST GRIA project [14]. It provides the necessary infrastructure for exposing computationally intensive applications across the Grid, with ancillary facilities for data staging and quality of service negotiation. A Grid service within GRIA can be considered as a *contextualised* web service, which exposes its functionality through a well-defined interface. It is contextualised since interactions with the web service are based on a well-defined process, with a context that is maintained throughout the lifetime of the process. It is the interaction protocols associated with these long-lived processes that we aim to make explicit through an appropriate formal description, so that they can be specified to an external access control and monitoring system.

In GRIA, a number of services and systems, both external and internal, are used. Internal systems and services include resource schedulers, accounting systems and databases, while external services include data staging services, certification authorities, and so forth. GRIA also provides features such as negotiation over the quality of service and long-term accounts with service providers. We do not discuss these issues in detail here, but the interested reader is referred to [14], in which a more complete description of the GRIA system is available.

Rather, what we present here is a simplified example of the operation of GRIA, and a description of how these concepts are mapped to an electronic institution. Our example is based on a straightforward usage scenario for Grid applications that is supported by GRIA. It involves a *client* that submits a computation job (such as rendering a short, animated video clip) to a *job service*, where the computation job specifies a particular application to execute, such as a renderer. Now, in order for a client to be able to submit a computation job it must first have an *account* open where the computation job is able to bill for services. Furthermore, it must have the resources of the computation job allocated to it via a *resource allocator*.

In typical Grid scenarios, accounts are opened by *Budget Holders* (e.g. the manager of a research group), who then allow *Account Users* (e.g. the individual researchers planning and running jobs) access to the account so that they can allocate resources and run jobs charged to the account, etc.

The main limitations of the current GRIA implementation are as follows:

- Currently, the service interaction model is fixed as a static factory pattern. The business processes linking the *Account Service*, *Resource Allocation* and *Job Service* cannot be changed to fit local policies or business models.
- The interactions between services are encoded through a shared state held within the services themselves. This means that services cannot exist in different domains. While it is entirely reasonable for the *Resource Allocation* service to be collocated with the *Job Service* that uses its resources, it should not be necessary for the *Account Service* also to be operated by the same domain.
- There is no explicit description of the service interactions. This means that one cannot provide any external monitoring to detect any corruption of the services, which might become evident through some change in the interaction with them.

3 A Desired Scenario

Consider the collection of services and service clients shown in Figure 1, which illustrates the example described above. In this figure, we represent the different web services involved, whereas the functional statements positioned above the arrows represent the methods that could be used to interact with the services on the right of the organisational boundary.

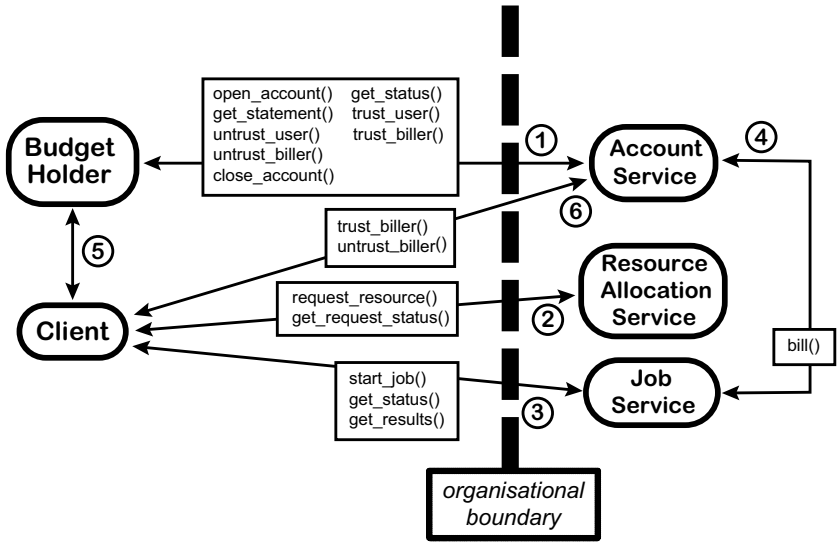

Fig. 1. Grid Interaction Example

This scenario is based heavily on GRIA, but significantly simplified to make it clear and tractable enough for our purposes. However, in one way, Figure 1 is more sophisticated than the current GRIA implementation: some interactions that would be hidden in the "back office" within a GRIA deployment have been included in the service interfaces, so that we can construct a scenario in which the *Account Service* is not collocated with the *Resource Allocation Service* and the *Job Service*.

The interactions between clients and services are as follows:

- A *Budget Holder* is able to interact with the *Account Service* (1). It first requests the account to be opened and, once the account is active it can, amongst other actions, delegate or revoke access to the account by account users and allow billers to charge for their services to the account.
- A *Client* is able to interact with the *Resource Allocation* service (2) so as to request access to a computation services.
- Once a resource has been allocated, the *Client* can interact with the *Job Service* (3), requesting the computation to be run.

– Before starting to run a job, the *Job Service* must be able to charge, or bill some entity for performing the job. The *Job Service* does this by getting a contextualised endpoint for the *Account Service* (4) representing an Account, and billing the Account for the job using an operation of the specified *Account Service* in the specified context.
– The contextualised endpoint for billing the Account must be obtained from the *Account Service*. To get one, the *Client* must be authorised by the *Budget Holder* (5), who must call an operation of the *Account Service* (6) to inform it of the *Client's* trusted status.
– In the case where the account credit has run out, or the account has been closed, all *Account Users* should not be allowed to initiate any further resource allocations or jobs. However, it should still be possible for *Billers* to bill for any outstanding jobs remaining until the account has been properly cleared.

In trying to describe these interactions, we must also take note of other more practical challenges.

– Some interactions, such as the opening of an account, are lengthy processes that necessarily involve both online and offline actions. For example, an *Account Manager* may need to perform credit checks offline before approving a *Budget Holder's* account.
– It is likely that the *Budget Holder* and *Client* are behind opposed conventional firewalls. Bearing in mind that on the Grid, interactions may persist for a long time, this means all interactions must be initiated by clients, because if the services try to do so, their attempt may be blocked by the client-organisation's firewall.

This second point means that the interactions are one-sided, with clients polling services for the current status of the interaction where necessary. For example, a *Budget Holder* should be able to poll the *Account Service* to find out when their account has been approved, and the *Client* must poll the *Job Service* to find out when a job starts or terminates. In the context of an agent-oriented approach to modelling this scenario, we note that there are services that cannot initiate interactions. This is different to the more general agent models, in which agents are both proactive and reactive.

4 Semantic Firewall

Our goal is to enhance security in a services-oriented environment whilst addressing the challenges and limitations described above. We aim to decouple services by providing well-defined interaction protocols, and eliminating the need for the services to deal with undesired messages by filtering out such messages at the organisational level. Furthermore, we want to provide network administrators with the ability both to allow flexible interaction with Grid services (something not possible using conventional firewall technologies) and to maintain careful control over those interactions.

To achieve these goals, we introduce the notion of a security device which is able to reason about the current state of interaction between external services, and those services protected by the security device, and also to ensure that all messages sent to

these services are consistent with the current state. We use the term *Semantic Firewall (SFW)* to describe the device since, as opposed to a normal firewall, it monitors traffic at the level of messages exchanged between web services and takes into account the context of interaction. It is important to emphasise that the SFW is only concerned with, and protects, the *interests* of the protected service, and thus does not require a *global view* of all the interactions taking place within the context of a client attempting to achieve a task in which the protected service is also involved. For example, in the above example, the SFW does not need to be aware of the interactions between the *Budget Holder* and the *Client*.

The requirements for the SFW are divided into *description and reasoning*, and *infrastructure* requirements. The former refers to what we should be able to describe about the services and interactions between them and what type of reasoning we should be able to perform, while the latter refers to what the infrastructure should be able to do given the descriptions and reasoning over them.

1) Description and Reasoning Requirements

Allowed Participants: The first step is for the SFW to have an appropriate set of descriptions of what entities are allowed to interact with protected services, and for the SFW to be able to appropriately identify the services attempting to communicate with protected services. In part, the solution involves the use of "conventional" security technologies such as PKI and X.509 for user authentication. However, beyond such technologies we must also look at the *context* of interaction and the intent of the interaction, which is an issue that the SFW, rather than lower-level security technologies, will handle.

Allowed Interactions: Subsequently, based on who is attempting to interact, we require a description of a currently permissible interaction protocol. The possible interactions in a web services environment are based on the methods described within the WSDL (Web Services Definition Language)[4] interfaces for each service. However, WSDL interfaces do not provide any information about permitted processes for any given instant. Instead, developers typically rely on documentation associated with the services to determine the appropriate process through which methods in the WSDL interface should be called. Our aim is to ensure that this process is adhered to, by providing the SFW with the descriptions of the process.

Dependencies between parties: The SFW must be aware of the dependencies between interaction protocols for different parties. This includes both the manner in which actions from one party can *limit* what another party can do, and how actions from one party can *enable* another to interact with a protected service.

2) Infrastructure Requirements

Transparent protection: The infrastructure should take into account the fact that the SFW should be invisible to services outside the protected domain. Whilst we may foresee a future situation in which several SFWs, each operating within a different organisational domain, play an active part in defining and supporting the context through

[4] http://www.w3.org/TR/wsdl

which services from those domains can interact, we must begin with the assumption that external services are not aware of the existence of such a device.

Informing users on reasons for failure: In order for both system administrators and users to accept any actions taken by the SFW (such as rejecting messages, etc), the device should be able provide justifications about its actions, such as why an interaction was accepted or rejected. A clear trace of the reasoning of the device is necessary to achieve this requirement.

5 Electronic Institutions

Given the set of requirements described in the previous section, an essential component is the existence of an interaction protocol and a means of defining the protocol and its dependencies. Although there are a variety of technologies that enable us to define interaction protocols (e.g. [11, 1]), as well as a significant amount of work on describing appropriate policies [15], what we require is something that take a more integrated view of the situation. In this regard, EIs are able to address several of the concerns raised above. Below we provide a brief overview of this work before moving on to describe how the concepts of Electronic Institutions can be mapped to those in GRIA, so as to provide appropriate descriptions that the SFW can use to monitor interactions.

To define an EI, it is necessary first to define a common language to allow agents to exchange information, the activities that agents may perform within the institution, and the consequences of their actions. Our model of electronic institutions is thus based on four principal elements: a dialogical framework, a set of scenes, a performative structure and a set of normative rules [3, 10, 12].

The *dialogical framework* defines the valid illocutions that agents can exchange, and the participant roles and relationships. In the most general case, each agent that exists within in a multi-agent environment is endowed with its own inner language and ontology. In order to allow agents to successfully interact with others we must address the fundamental issue of relating their languages and ontologies to each other. EIs solve this problem simply by establishing acceptable illocutions, communication primitives and knowledge representation concepts through a common, well defined ontology (vocabulary) — the common language to represent the "world" — that all the agents adhere to. Moreover, the dialogical framework defines the participant roles within the EI and the relationships among them. Each role defines a pattern of behaviour within the institution, and any agent within an institution is required to adopt a subset of them. In the context of an EI, we distinguish between two types of roles, *internal* and *external* roles. The internal roles can only be played by what we call *staff* agents which are those pertaining to the institution. These are analogous to workers within human institutions. Since an institution delegates their services and duties to the internal roles, an external agent can never play an internal role. By sharing a dialogical framework, we enable the heterogeneous community of agents to exchange knowledge with each other.

The set of possible activities within an electronic institution is defined by the composition of multiple, distinct, and possibly concurrent dialogic activities, where each activity involves different groups of agents playing different roles. For each activity, interactions between agents are articulated through agent-group meetings, which follow

well-defined communication protocols; we refer to such meetings as *scenes*. Thus, all agent interactions that take place within an EI exist within the context of a scene. In addition, the protocols for each scene model the possible dialogic interactions between *roles* instead of *agents*; thus, scene protocols define patterns of multi-role conversation, and hence can be multiply instantiated by different groups of agents. A distinguishing feature of scenes is that they allow agents either to enter or to leave a scene at certain particular moments (states) of an ongoing conversation depending on their role.

A scene protocol is specified by a directed graph, where the nodes represent the different states of the conversation, and the arcs are labelled with illocution schemes or timeouts that allow the conversation state evolve. Thus, at each point of the conversation, the EI defines what can be said, by whom and to whom. As we want the protocol to be generic, state transitions cannot be labelled by grounded illocutions. Instead, illocution schemes have to be used where, at least, the terms referring to agents and time must be variables, whilst other terms may be either variables or constants. Thus, the protocol is independent of concrete agents and time instants. Moreover, arcs labelled with illocution schemes can have some associated constraints which impose restrictions on the valid illocutions, and on the paths that the conversation can follow.

While a scene models a particular multi-agent dialogic activity, more complex activities can be specified by establishing relationships among scenes, captured in the *performative structure*. In general, the activity represented by a performative structure can be depicted as a collection of multiple, concurrent scenes. Agents navigate from scene to scene, constrained by the rules defining the relationships among scenes. In order to capture the relationships between scenes, we use a special type of scene, known as *transitions*. Transitions allow the expression of agent synchronisation points (i.e. selection points where agents can decide which path to follow), or parallelisation points (i.e. where agents are sent to more than one scene). They can be seen as a type of router in the context of a performative structure. Moreover, the very same agent can possibly participate in multiple scenes at the same time. Likewise, there may be multiple concurrent instantiations of a scene, so we must also consider: 1) whether the agents following the arcs from one scene to another are allowed to start a new scene execution; 2) whether they can choose to join just one or a subset of the active scenes; or 3) whether they can choose to join all active scenes.

A performative structure can be seen as a network of scenes in which their connections are mediated by transitions that determine the role flow policy. Finally, from the set of scenes, the initial and final scenes determine the entry and exit points of the institution respectively.

In the context of an institution, agent actions have consequences, usually in the shape of compromises which impose obligations or restrictions on dialogic actions of agents in scenes in which they are acting (or will be acting in the future). Normative rules affect the behaviour of agents by imposing obligations or prohibitions.

Note that we are considering dialogic institutions, and the only actions considered are the utterance of illocutions. Therefore, we can refer to the utterance of an illocution within a scene or when a scene execution is at a concrete state. The intuitive meaning of normative rules is that if illocutions are uttered in the corresponding scene states

(and some predefined expressions are satisfied), then other illocutions satisfying other expressions must be uttered in the corresponding scene states.

To summarise, the notions presented above define the regulatory structure of an EI as a "workflow" (i.e. performative structure) of multi-agent protocols (scenes) along with a collection of (normative) rules that can be triggered off by an agent's actions (speech acts).

Note also that the formalisation of an EI focuses on macro-level (societal) aspects, instead of on micro-level (internal) aspects of agents. This allows us to more easily map the concepts between Grid environments and EIs. Since no assumptions are made about internal aspects of agents, it is possible to define one-to-one mappings between actions (or services) provided by each agent, and web services defined within a Grid environment.

6 Using Electronic Institutions in GRIA

Given the descriptions of the requirements for the Semantic Firewall in section 4 and the overview of the main Electronic Institution concepts in section 5, it is now possible to investigate how such concepts can be applied within the SFW. This is achieved by defining each *scenario* of interaction with the protected domain as an Electronic Institution. A scenario will typically be associated with a specific business model, such as described in Section 3.

6.1 Mapping GRIA Models to Electronic Institutions

Services: Each service that is expected to interact in a well-defined scenario with a protected service is associated with a role within the electronic institution. *External roles* are used to represent services that are not protected by the SFW, whereas *internal roles* are used for the protected services. This allows us to clearly distinguish between those services that perform institutional services and that the SFW has a responsibility of protecting, and external services that may be providing the client with a service but do not form part of the institution. In our running example, the *Account Service*, *Resource Allocation* and *Job Service* occupy internal roles, whereas the *Budget Holder* and *Client* occupy external roles.

Interactions and Business Process: The allowed interactions between services and the entire business process can be encoded as individual scenes within an EI. The participants in the scene are the relevant services, whilst the illocutions being uttered are mapped to the corresponding WSDL methods. In addition, in those situations where the SFW itself needs to be made aware of events that occur within protected services, it appears as a participant within a scene. To illustrate this, consider the case where a request is sent by a *Budget Holder* to close the account. In this case, the *Account Service* may still allow *Billers* to bill the account up to the point where the account has been settled (which may involve offline actions). When the account is finally closed, the SFW needs to informed about this closure explicitly by the *Account Service*, since

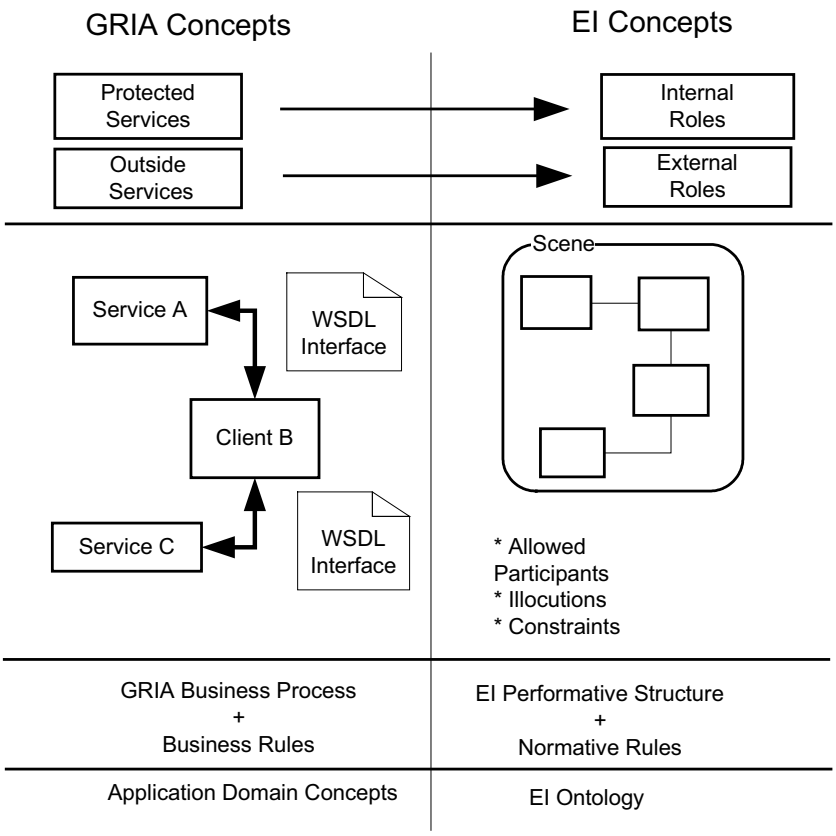

Fig. 2. Mapping GRIA concepts to EIs

there is no illocution that will enable it to understand that. In this case the SFW is an active participant in the scene.

The wider business process, with regards to a particular task and the protected services, is described by the performative structure of the electronic institution. This allows us to define the appropriate flow of roles between scenes as well as impose a particular process or workflow to the entire set of interactions with different parties.

Cross-party dependencies: We have already mentioned that an important goal of the SFW is that of managing the dependencies between interacting parties. Returning to the example mentioned above, once a *Budget Holder* has requested that an account should be closed, access should be restricted to all clients associated with the closed account to prevent them from assigning other billers. Thus, an action within a scene that involves both the *Budget Holder* and the *Account Service* also has an implication on the permissible actions within scenes involving the *Account Service* and *Clients*.

Within an EI, this can be modelled by defining a set of norms, to ensure that specific actions can hold only as long as some constraints hold true.

Domain Ontology: The application domain concepts that are relevant to the interactions between protected and external services are encoded within the EI ontology. The ISLANDER editor supports the management of such ontologies, thereby facilitating the creation of mappings between the datatypes used within the EI definition and the datatypes used by the web service interface.

6.2 Semantic Firewall Core Modules

Given the discussion of the mapping between the EI concepts and SFW concepts in the previous section, it is now proceed to address the structure of the SFW itself, illustrated in Figure 3.

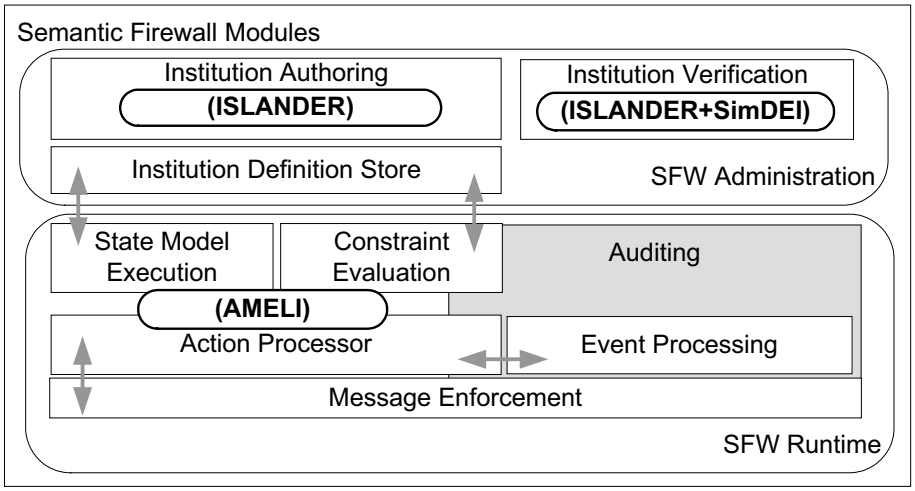

Fig. 3. Semantic Firewall Core Modules

The SFW has two main components: the *Administration* and the *Runtime Environment*. The SFW Administration deals tasks such as authoring, verification and storage of electronic institutions, whereas the SFW Runtime Environment is responsible for the verification of messages based on the electronic institution definitions. We discuss each of these in more detail below.

Semantic Firewall Administration: SFW Administration is divided into three different modules:

– *Authoring:* The ISLANDER tool provides a graphical interface to facilitate the definition of an institution. It allows for the definition of a common ontology, the performative structure and related scenes, as well as related norms.

– *Verification:* For verification of the electronic institution, ISLANDER can provide verification of the *structural properties* while verification of the dynamic behaviour can be achieved through simulation in the SIMDEI tool [13].
– *Storage:* A verified definition of the SWF is stored in the *Institution Definition Store* for use by the SFW Runtime.

Semantic Firewall Runtime: The SFW Runtime consists of several modules, and is primarily concerned with the verification of each message passing to protected services.

– *The Message Enforcement Module:* This is responsible for receiving messages and dealing with all lower level issues, such as parsing the SOAP structure of messages and providing the relevant part of the message to the *Action Processor*, which performs the mapping between the WSDL message and the definition within the electronic institution.
– *The State Model Execution and Constraint Evaluation Modules:* These modules are queried to determine whether the message is a valid one based on the electronic institution definition. This functionality can be provided by the AMELI run-time engine [5] which can directly accept a definition of an EI and can reason about what are the next allowable steps according to the definition.
– *The Event Processing Module:* At the same time as the *State Model Execution* and *Constraint Evaluation* modules are being queried, the *Event Processing* module collects information sent by the protected services to the SFW, whenever that is appropriate as discussed earlier.
– The Auditing Module: This module is responsible for keeping a record of the various actions so provide a trace as to why messages may have been rejected.

7 Evaluation Case Study

In order to better illustrate the use of Electronic Institutions within the SFW, this section presents a case study which includes a description of the performative structure, followed by a simplified definition of the scene dealing with account management.

Figure 4 illustrates the GRIA business process as an electronic institution's performative structure. It contains a collection of scenes (represented as boxes) relating to each of the GRIA services[5]. We differentiate between *internal roles* representing the GRIA protected services; in this case the services behind the organisation boundary as depicted in Figure 1, *Account Manager* (AM), *Job Manager* (JM), *Resource Manager* (RM), and *Semantic firewall* (SF); and *external roles* representing the GRIA external users, which include the *Budget Holder* (BH), *Account User* (AU), and *Job User* (JU). Agents playing these roles migrate from service to service after synchronising at transitions (represented by triangles).

Access to services is controlled through several scenes (see Table 1). All these scenes are specified to realise a client-server model. Thus, for instance, when a *JU* agent

[5] Note that the connections between scenes are labelled with the roles migrating from service to service along with agent variables that are expected to be bound to actual agent identifiers at run-time.

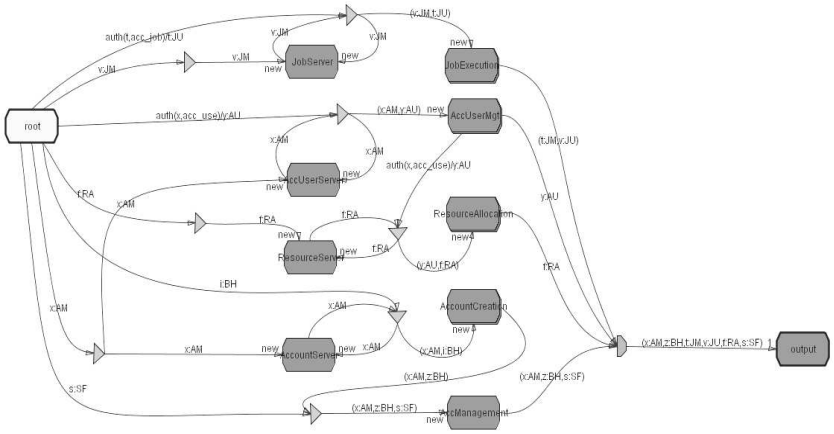

Fig. 4. GRIA Performative Structure

requires a job execution, it first synchronises with a *JM* agent that is continuously listening to agents' requests at the `JobServer` scene. Thereafter, the two agents progress together through the transition to create a new execution of the `JobExecution` scene. Note that the scenes offering the GRIA protected services are specified so that they can be multiply instantiated, and thus serve multiple agents' requests simultaneously. Note also that there are scenes (`JobServer`, `ResourceServer`, `AccountServer`, and `AccUserServer` particularly devoted to the listening functions of the agents playing the internal roles.

Service	Offered through Scene
Resource Allocation Service	`ResourceAllocation` Scene
Job Service	`JobExecution` Scene
Accounting Service Service	`AccountCreation` Scene `AccUserMgt` Scene `AccManagement` Scene

Table 1. Various services offered through different scenes

Next, we examine the `AccManagement` scene, illustrating how the specific interactions with services are managed. There are three participant roles in this scene, the *Account Manager (AM)* represented by the *Account Service*, the *Budget Holder (BH)* and the *Semantic Firewall (SF)*. The boxes represent different states of the dialog, while the arcs between them represent possible illocutions.

At `W0` all roles are allowed to enter the scene. At this state the *BH* is allowed to request a statement of the account (`arc 0`), to which the *AM* can reply with a statement. In addition, the *BH* can request for the *AM* to trust a biller (`arc 1`), which the AM can either acknowledge positively (`arc 2`) or refuse (`arc 5`). If the request is accepted this will enable a *Client*, matching the criteria of the user that should be trusted to enter the institution and also assign billers to this account, as we discussed in Section 3. The *BH* can also request for an account to be closed (`arc 4`). This lead the scene to a state

where the only thing the *BH* can do is request the status of the account and once the account has been closed, which as we already mentioned may involve offline actions, the *SF* is informed of this so that is can reflect this change on the allowed actions of other interested parties such as billers.

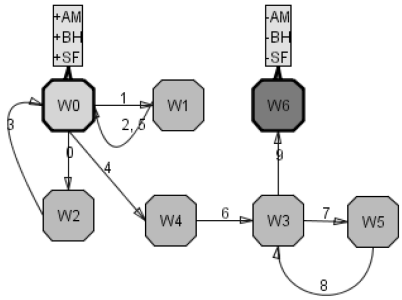

Fig. 5. Account Management Scene

8 Conclusion

In this paper we proposed a method for enhancing security within Grid environments by making use of Electronic Institutions to support the specification, verification and monitoring of permissible interactions within a protected (i.e. firewalled) environment. This is achieved though a dedicated device, the *Semantic Firewall*, which maintains a set of mappings between entities within Electronic Institutions and Grid Services. The Semantic Firewall facilitates the integration of agent technologies within a Grid environment, without requiring radical changes to the infrastructure or the way developers build Grid services. As such, this work represents a pragmatic example of how the worlds of Grid infrastructure and agent research can come together to provide effective solutions to the existing limitations for Grid infrastructure.

The work described in this paper provides several avenues for further development. In the short-term, we can begin to define more flexible business models within GRIA, since we can take advantage of the flexible description and monitoring capabilities to ensure that they are adhered to. Subsequently, we can begin to examine how such institutions can be agreed upon at run-time between different organisations, where each protected by a Semantic Firewall. Finally, we must also begin to investigate the possibility of making *deployment* of services within a Grid environment more flexible by providing high-level definition of allowed processes (as EIs) which developers can then ensure they adhere to.

9 Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Semantic Firewall project (ref. GR/S45744/01).

References

1. R. Ashri, G. Denker, D. Marvin, M. Surrdige, and T. R. Payne. Semantic Web Service Interaction Protocols: An Ontological Approach. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Int. Semantic Web Conference*, volume 3298 of *LNCS*, pages 304–319. Springer, 2004.
2. K. Czajkowski, D. F. Ferguson, Foster I, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. The WS-Resource Framework. Technical report, The Globus Alliance, 2004.
3. M. Estena. *Electronic Institutions: from specification to development*. PhD thesis, Technical University of Catalonia, 2003.
4. M. Esteva, D. de la Cruz, and C. Sierra. ISLANDER: an electronic institutions editor. In *The First Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 1045–1052. ACM Press, 2002.
5. M. Esteva, J. A. Rodriguez-Aguilar, B. Rosell, and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions pages 236-243, new york, usa, july 19-23 2004. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 236–243. ACM Press, 2004.
6. I. Foster. The Anatomy of the Grid: Enabling Scalable Virtual Organisations. In R. Sakellariou, J. Keane, J.R. Gurd, and L. Freeman, editors, *7th International Euro-Par Conference*, volume 2150 of *LNCS*. Springer, 2001.
7. I. Foster, N. R. Jennings, and C. Kesselman. Brain meets Brawn: Why Grid and Agents need each other. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 8–15. ACM Press, 2004.
8. I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.
9. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, 35(6):37–46, June 2002.
10. P. Noriega. *Agent Mediated Auctions: The Fishmarket Metaphor*. PhD thesis, Technical University of Catalonia, 1997.
11. S. Paurobally, J. Cunningham, and N. R. Jennings. Developing Agent Interaction Protocols Using Graphical and Logical Methodologies. In M. Dastani, J. Dix, and A. El Fallah-Segrouchni, editors, *PROMAS*, volume 3067 of *LNCS*, pages 149–168. Springer, 2003.
12. J. A. Rodriguez-Aguilar. *Towards a Test-bed for Trading Agents in Electronic Auction Markets*. PhD thesis, Technical University of Catalonia, 2001.
13. C. Sierra, J. A. Rodriguez-Aguilar, P. Noriega, M. Esteva, and J. L. Arcos. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professionall*, 4, 2004.
14. S. Taylor, M. Surridge, and D. Marvin. Grid Resources for Industrial Applications. In *2004 IEEE Int. Conf. on Web Services (ICWS'2004)*, 2004.
15. G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic web languages for polic representation and reasoning: A comparison of kaos, rei and ponder. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proceedings of the 2nd International Semantic Web Conference*, volume 2870 of *LNCS*, pages 419–437. Springer, 2003.
16. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open grid services infrastructure. Technical report, Global Grid Forum, 2003.
17. M. J. Wooldridge and N. R. Jennigns. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3):20–27, 1999.