REPUTATION MANAGEMENT IN GRID-BASED VIRTUAL ORGANISATIONS

Alvaro Arenas, Benjamin Aziz
e-Science Centre, STFC Rutherford Appleton Laboratory, Oxfordshire OX11 0QX, UK
A.E.Arenas@rl.ac.uk, B.Aziz@rl.ac.uk

Gheorghe Cosmin Silaghi
Babes-Bolyai University, Faculty of Economics, Cluj-Napoca, Romania
gsilaghi@econ.ubbcluj.ro

Keywords: Reputation Management; Grid computing; Virtual Organisations.

Abstract: Grid computing allows one to access, utilise and manage heterogeneous resources in virtual organisations across multiple domains and institutions. The formation and operation of virtual organisations involve establishing trust among their members and reputation is one measure by which such trust can be quantified and reasoned about. This paper presents a reputation model for Grid-based virtual organisations that can be used to rate users as well as resources and their providers. The model is based on utility computing, which expresses the satisfaction of an entity in its interactions with other entities with respect to some issues of interest. We show how the model can be used to rate users according to their resource usage and resource providers according to the quality of service provided. Finally, we demonstrate through Grid simulations, how the model can be useful in improving completion and welfare in a virtual organisation.

1 INTRODUCTION

A Virtual Organisation (VO) can be seen as a temporary or permanent coalition of geographically dispersed organisations that pool resources, capabilities and information in order to achieve common goals. VOs are given attention by researchers within a wide range of fields, from social anthropology and organisational theory to computer science. Their importance resides in providing an abstraction to represent organisational collaborations, a topic of fresh interest given the current exploitation of Internet technology to create virtual enterprises, or the sharing of resources across different organisations as envisaged by Grid computing (Foster et al., 2001).

This paper investigates how to exploit reputation systems for managing Grid-based VOs. Reputation is one measure by which trust among different members of a VO can be quantified and reasoned about. We focus on Grids where the availability of resources and user tasks is highly dynamic, and both resource providers and users have to compete for providing and employing resources. Reputation systems are then used to manage reputation of resource providers, according to the Quality of Service (QoS) provided, as well as reputation of users, according to their usage of resources.

The main contribution of the paper is to study the impact of applying reputation in Grid-based VOs. Resource-providers reputation can be used by resource brokers in order to improve allocation of user tasks by selecting reputable providers. Conversely, users reputation can be used by resource providers in order to define security level for users; low-reputable users would be assigned tight measures when accessing a resource.

The structure of the paper is the following. Section 2 describes our model of VOs. Then, section 3 introduces the reputation model used in the paper, a utility-based model that uses information provided by monitors to rate entities within a Grid. This reputation model is used in section 4 to model resource-providers and user reputations. Next, section 5 presents the system architecture and gives an example of a usage scenario. Then, section 6 shows experimental results and discusses the use of reputation for both brokering and controlling resource usage. Section 7 reviews related work and finally, section 8 concludes the paper and highlights future work.

2 A MODEL OF VIRTUAL ORGANISATIONS

In order to support rapid formation of VOs, we use the concept of *virtual breeding environment* (VBE) (Camarihna-Matos and Afsarmanesh, 2003). A VBE can be defined as an association of organisations adhering to common operating principles and infrastructure with the main objective of participating in potential VOs. In this paper, we have adopted the view that organisations participating in a VO are selected from a VBE, as illustrated in figure 1. Such organisations may provide resources, represented by ovals, and include users that utilise VO resources, represented by small squares. Organisations pre-register to a VBE

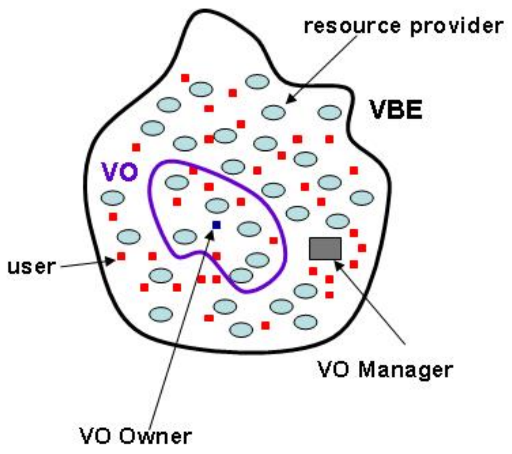

Figure 1: VBE and VO Models

via the VO Manager component, including description of the resources they are willing to share in a Grid and the list of potential users belonging to the organisation. When a user requires to create a VO, he assumes the role of VO Owner and contacts the VO Manager with the description of needed resources. The VO Manager includes a resource broker component that select potential resource providers. The VO Owner then selects resource providers and define list of users of the VO.

Note that a VBE can be seen as a market place, where resource providers are competing to participate in a VO, and users within a VO are competing to use resources. Reputation information about resource providers can be used as another parameter for guiding the selection of VO partners. On the other hand, having reputation information about users could help resource providers to implant tighter security mechanisms for accessing resources. Next sections would develop these ideas for Grid-based VOs.

3 UTILITY-BASED REPUTATION MODEL

The reputation model presented here is based on the reputation model described in (Silaghi et al., 2007). This section summarises the main properties of such a model and adapts it for deadling with Grid-based VOs.

Central to our model is the notion of organisation; the set of all organisations is denoted by *Org*. From the perspective of the reputation model, a VO is represented just as a set of organisations; we keep track of all VOs that have existed and use the set *VOId* to denote the set of all VO identifiers.

The *entities* we want to keep reputation values are defined as elements of the set *Ent*. An obvious restriction is that an entity must belong to an organisation. We are interested in some particular *issues of interest* associated to an entity; the set of all issues of interest is represented by *Issue*. The individuals that use (*consume*) the entities and qualify them are members of the set *Cons*. The sets *Ent*, *Issue* and *Cons* are considered type parameters that will be instatiated according to the domain we are interested in. Below we represent above types and their associations as functions, using the mathematical notations provided by the Z specification language (Woodcock and Davies, 1996).

$[Org, VOId]$

$$VOS : VOId \nrightarrow \mathbb{P}\, Org$$

$[Cons, Ent, Issue]$

$$\begin{array}{l} EntOrg : Ent \rightarrow Org \\ EntIssue : Ent \rightarrow \mathbb{P}\, Issue \\ EntCons : Ent \nrightarrow \mathbb{P}\, Cons \end{array}$$

Notation $[Org, VOId]$ introduces *Org* and *VOId* as basic types. *VOS* associates a VO with the set of organisation participating in it. Function *EntOrg* associates an entity with the organisation to which belongs to; *EntIssue* relates an entity with its issue of interests; and *EntCons* associates an entity with its consumers.

In our model, it is assumed the existence of monitors that deliver events indicating the current value produced by an entity for a consumer in relation to a particular issue of interest within a VO. We represent an event as a tuple that contains at least four elements: a *consumer*, an *entity*, an *issue*, and a *VO*. A trace correspond to a sequence of events:

$$\begin{array}{lcl} Event & == & Cons \times Ent \times Issue \times VOId \times \cdots \\ Trace & == & \mathrm{seq}\, Event\,. \end{array}$$

A utility function reflects the satisfaction of a consumer in relation to particular entity. It relates an event with a numeric value indicating what is really expected by the consumer:

$$utility : Event \rightarrow \mathbb{R}\ .$$

The complete definition of a utility function is considered to be domain specific.

Utility functions are used to define the reputation of an entity in relation to a particular issue of interest from the perspective of a consumer.

$$[Cons, Ent, Issue]$$
$$rep\_eic : (Cons \times Ent \times Issue \times VOId) \rightarrow \mathbb{R}$$
$$\forall c : Cons, e : Ent, i : Issue, vo : VOId \bullet$$
$$rep\_eic(c,e,i,vo) = \frac{\sum_{ev \in Trace \upharpoonright \{(c,e,i,vo,\cdots) \in Event\}} utility(ev)}{\#(Trace \upharpoonright \{(c,e,i,vo,\cdots) \in Event\})}$$

where $\#s$ denotes the cardinality of sequence s and $s \upharpoonright A$ denotes the largest subsequence of s containing only those objects that are elements of A. Reputation *rep_eic* is defined as the average of the utilities obtained from all generated events so far; it is defined as a generic function paremeterised by sets $Cons, Ent$ and *Issue*.

Aggregating all consumers reputation about an entity within a VO produces the reputation of the entity in the VO. Let us first define the reputation of an entity for the case of a particular issue of interest in a VO as the aggregation of the reputation value given by each consumer in the VO.

$$[Ent, Issue]$$
$$rep\_ei : Ent \times Issue \times VOId \rightarrow \mathbb{R}$$
$$\forall e : Ent, i : Issue, vo : VOId \bullet$$
$$rep\_ei(e,i,vo) = \frac{\sum_{c \in EntCons(e)} rep\_eic(c,e,i,vo)}{\#EntCons(e)}$$

Reputation function *rep_ei* is defined as a generic function paremeterised by sets *Ent* and *Issue*.

The reputation of an entity in a VO is then the aggregation of the reputation of each of its issues of interest within the VO.

$$[Ent]$$
$$rep\_e : Ent \times VOId \rightarrow \mathbb{R}$$
$$\forall e : Ent, vo : VOId \bullet$$
$$rep\_e(e,vo) = \frac{\sum_{i \in EntIssue(e)} rep\_ei(e,i,vo)}{\#EntIssue(e)}$$

The reputation of an entity is an aggregation of its reputation in all VOs in which it has participated.

$$[Ent]$$
$$rep : Ent \rightarrow \mathbb{R}$$
$$\forall e : Ent \bullet rep(e) = \frac{\sum_{vo \in \operatorname{dom} VOS} rep\_e(e,vo)}{\# \operatorname{dom} VOS}$$

4 REPUTATION MODEL FOR GRID-BASED VIRTUAL ORGANISATIONS

This section describes how above reputation model can be used to keep reputation on both resource providers and users within a VO.

4.1 Reputation Management for Resource Providers

Here we aim at maintaining reputation for organisations (resource providers) within a VO and the VBE according to the quality of service (QoS) of the resources they provide. In this model, *consumers* correspond to the *users* in a VO, denoted by *VOUser*, and *entities* correspond to the *VO resources*, denoted by the set *Res*. There are several options for selecting issues of interest. We can have either a fine granularity where each service level objective defined for a resource can be seen as an issue of interest or a coarse granularity where the whole QoS can be seen as a single issue. For simplicity, we have selected the last option, having the whole QoS as our only issue of interest.

Functions *UsersVO* and *ResVO* represent the set of users of a VO and the resources that an organisation provides to a VO respectively.

$$UsersVO : VOId \rightarrow \mathbb{P}\, VOUser$$
$$ResVO : VOId \times Org \rightarrow \mathbb{P}\, Res$$

This model requires the existence in the system of monitors capable of detecting how the QoS varies for a resource and generating events to inform the reputation system about such changes. An event is then represented as a tuple denoting the current value of the QoS of a resource being used by a user within a VO.

$$Event \quad == \quad VOUser \times Res \times \{QoS\} \times VOId \times \mathbb{R}$$

where *QoS* is a name indicating the QoS issue.

In order to define the corresponding utility function, we introduce an auxiliary function indicating the service level agreement (SLA) accorded between a VO user and a resource provider for a particular resource within a VO.

$$SLA : VOUser \times Res \times VOId \rightarrow \mathbb{R}$$

The SLA function represents the *expected* quality of a resource. It is then used to define the utility (*satisfaction*) a user gets when employing a VO resource.

$$\begin{array}{|l} utility : Event \rightarrow \mathbb{R} \\ \hline \forall (u,r,QoS,id,v) \in Event \bullet \\ utility((u,r,QoS,id,v)) = \\ \qquad \begin{cases} 1 & \text{if } v \geq SLA(u,r,id) \\ \frac{v}{SLA(u,r,id)} & \text{if } v < SLA(u,r,id) \end{cases} \end{array}$$

We can now define the reputation of a resource using the reputation functions defined in the previous section. Here *Res_rep_eic* denotes the reputation value given by a particular VO user to a resource in relation to its QoS in a VO. *Res_rep_ei* represents the reputation of a resource taking into account its QoS in a VO; it is an aggregation of the reputation of all users of the resource in relation to the QoS issue within such VO. *Res_rep_e* denotes the reputation of a resource in a VO. Finally, *Res_rep* indicates the reputation of a resource in the VBE. Note, since we have only one issue of interest, *Res_rep_ei* and *Res_rep_e* are equivalent.

$$\begin{array}{lcl} Res\_rep\_eic & == & rep\_eic[VOUser, Res, \{QoS\}] \\ Res\_rep\_ei & == & rep\_ei[Res, \{QoS\}] \\ Res\_rep\_e & == & rep\_ei[Res] \\ Res\_rep & == & rep[Res] \end{array}$$

Using above functions, we can define the reputation of an organisation within a VO and in the VBE. The reputation of an organisation in a VO, denoted by *Org_rep_VO* can be defined as the aggregation of the reputation of all resources it provides to the VO.

$$\begin{array}{|l} Org\_rep\_VO : Org \times VOId \rightarrow \mathbb{R} \\ \hline \forall o \in Org, vo \in VOId \bullet \\ Org\_rep\_VO(o,vo) = \frac{\sum_{e \in ResVO(vo,o)} Res\_rep\_e(e,vo)}{\#ResVO(vo,o)} \end{array}$$

Reputation of an organisation in the VBE can then be defined as follows.

$$\begin{array}{|l} Org\_rep\_VBE : Org \rightarrow \mathbb{R} \\ \hline \forall o \in Org \bullet \\ Org\_rep\_VBE(o) = \frac{\sum_{vo \in \{v \in \mathrm{dom}\, VOS \mid o \in VOS(v)\}} Org\_rep\_VO(o,vo)}{\#\{v \in \mathrm{dom}\, VOS \mid o \in VOS(v)\}} \end{array}$$

4.2 Reputation Management for VO Users

Here we aim at maintaining reputation for users within a VO and the VBE according to their usage of VO resources. In this model, *users*, denoted by set *VOUser*, are seen as *entities* who could execute some pre-defined actions on resources following pre-established policies. *Resources*, denoted by set *Res*, are seen as *consumers* that qualify users in relation to their actions. If a user attempts to executate an action that is not allowed by the VO policy, it will be given a "bad qualification" by the resource that would be reflected in the user reputation.

Set *Action* denotes the set of possible actions to be perfomed on resources. A policy indicates the set of actions a user can perform on a resource in a VO. It represents the expected behaviour of a user.

$$\begin{array}{|l} policy : VOUser \times Res \times VOId \rightarrow \mathbb{P} Action \end{array}$$

A penalty function penalises a user with a value in the interval $[0,1)$ if he executes non-permitted actions

$$\begin{array}{|l} penalty : VOUser \times Res \times VOId \times Action \rightarrow [0,1) \\ \hline \forall u : VOuser, r : Res, vo : VOId, a : Action \bullet \\ \qquad (u,r,vo,a) \in \mathrm{dom}\, penalty \Rightarrow \\ \qquad\qquad a \notin policy(u,r,vo) \end{array}$$

Events are defined as follows

$$Event \quad == \quad Res \times VOUser \times \{Usage\} \times VOId \times Action$$

where *Usage* is a name indicating the resource-usage issue.

Functions *policy* and *penalty* are used to define the utility that a resource gets according to the actions performed by a user in a VO.

$$\begin{array}{|l} utility : Event \rightarrow \mathbb{R} \\ \hline \forall (r,u,Usage,vo,a) \in Event \bullet \\ utility((r,u,Usage,vo,a)) = \\ \qquad \begin{cases} 1 & \text{if } a \in policy(u,r,vo) \\ penalty(u,r,vo,a) & \text{if } a \notin policy(u,r,vo) \end{cases} \end{array}$$

We can now define the reputation of a user using the reputation functions defined in section 3. Here *User_rep_eic* denotes the reputation value given by a particular resource to a VO user in relation to the *Usage* of the resource in a VO. *User_rep_ei* represents the reputation of a user taking into account its resource usage in a VO; it aggregates the reputation of the user for all resources he uses in the VO. *User_rep_e* denotes the reputation of a user in a VO; it corresponds to an aggregation of the reputation of all his issue of interest. Since we have only one issue of interest, *User_rep_ei* and *User_rep_e* are equivalent. Finally, *User_rep* indicates the reputation of a user in the VBE.

$$\begin{array}{lcl} User\_rep\_eic & == & rep\_eic[Res, VOUser, \{Usage\}] \\ User\_rep\_ei & == & rep\_ei[VOUser, \{Usage\}] \\ User\_rep\_e & == & rep\_ei[VOUser] \\ User\_rep & == & rep[VOUser] \end{array}$$

5 A REPUTATION MANAGEMENT SYSTEM

Here we present the architecture of a VO that uses reputation management in order to facilitate the rating of both VO resources and VO users. The architecture is being implemented in the EU FP6 project GridTrust [1]. In this architecture, the VO Management subsystem consists of other services in addition to the *Reputation Management* (RM) service, such as a *VO Manager* (VOM), a *Reputation-aware Resource Broker* (RRB) and a *Resource Usage Control and Monitoring* (RUCM) service. These services are shown in Figure 2.

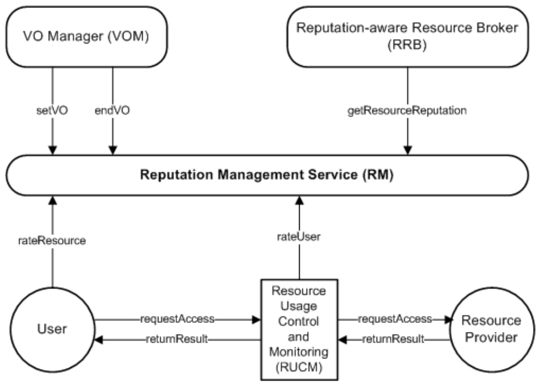

Figure 2: Reputation Management in VOs

The VOM informs the RM service of the setting up of a new VO, which includes the registration of the list of VO resources and users, as shown in the following protocol:

VOM→ RM : setVO(VO ID, Resource ID List, User ID List)
RM→ VOM : ack()

Where *VO ID* is the identity of the VO being registered, *Resource ID List* is the list of resources of the VO, and *User ID List* is the list of users in the VO. On the other hand, the termination of an existing VO is carried out through the following protocol:

VOM→ RM : endVO(VO ID)
RM→ VOM : ack()

Where *VO ID* is the identity of the VO being terminated. The RRB service is used during the setting of new VOs by the VOM. During this phase, the RRB may request from the RM service the reputation of a resource in a particular VO or in the general VBE before proposing it to the VOM:

RRB→ RM : getResourceRep(Resource ID, VO ID)
RM→ RRB : return(Resource ID, Reputation Value)

[1]https://www.gridtrust.eu

In case the VO ID is assigned a NULL value, the returned reputation will be the resource's reputation in the general VBE.

The RUCM service is a service that monitors requests and replies sent to and from a resource in its interaction with a VO user. The RUCM service can detect any undesirable behaviour by the user in its usage of the resource being protected by that instance of the RUCM service. This could be for example the excessive storage of data on the resource beyond the user's quota. Hence, the RUCM service can report prohibited actions performed by the VO users to the RM service as follows:

RUCM→ RM : reportUser(Resource ID, User ID, VO ID, Action)
RM→ RUCM : ack()

The RM service can also accept ratings by the VO users of the QoS levels they have experienced in their interactions with VO resources. This is done through the following protocol, in which the user reports the QoS value:

User→ RM : rateResource(User ID, Resource ID, VO ID, QoS Value)
RM→ User : ack()

Finally, any of the entities in a VO may request from the RM service the reputation of a user:

Any→ RM : getUserRep(User ID, VO ID)
RM→ Any : return(User ID, Reputation Value)

Again, in the event that the VO ID is assigned a NULL value, the returned reputation will be the user's reputation in the general VBE.

5.1 Usage Scenario

We consider here an example of a usage scenario of the RM system as shown in Figure 3. We assume that a RRB starts by querying the RM system for the reputation of a couple of resources, Resource1 and Resource2, in order to join them to a new VO. After that, the VOM signals to the RM system the setting up of the new VO and informs the latter of the two resources and three users, User1, User2 and User3. Once this operation is acknowledged by the RM system, the VO becomes operational and the users can avail of the resources offered.

At some stage, the RCUM service at Resource1 captures a prohibited action performed by User3 and thus reports it to the RM system. Based on the utility function for Resource1 and the penalty for the prohibited action, the RM system computes the satisfac-

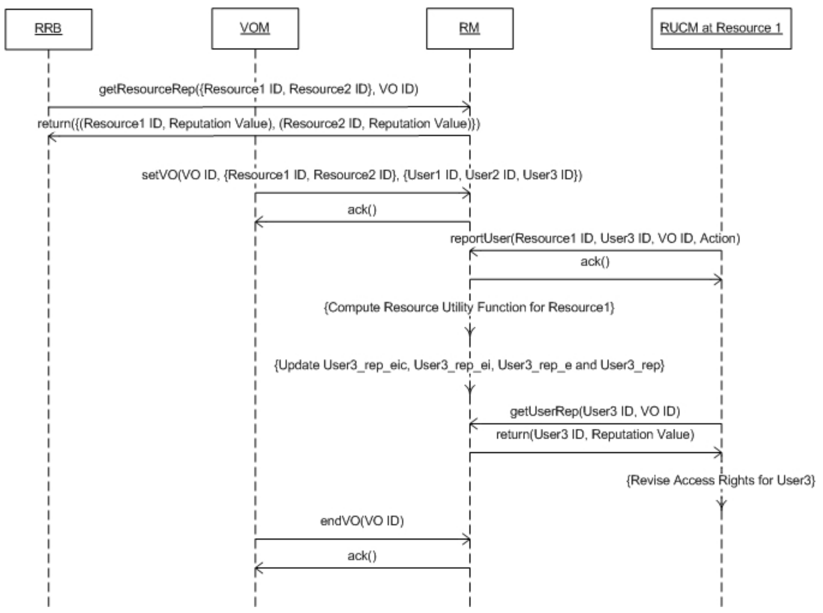
RRB
VOM
RM
RUCM at Resource 1
getResourceRep({Resource1 ID, Resource2 ID}, VO ID)
return({(Resource1 ID, Reputation Value), (Resource2 ID, Reputation Value)})
setVO(VO ID, {Resource1 ID, Resource2 ID}, {User1 ID, User2 ID, User3 ID})
ack()
reportUser(Resource1 ID, User3 ID, VO ID, Action)
ack()
{Compute Resource Utility Function for Resource1}
{Update User3_rep_eic, User3_rep_ei, User3_rep_e and User3_rep}
getUserRep(User3 ID, VO ID)
return(User3 ID, Reputation Value)
{Revise Access Rights for User3}
endVO(VO ID)
ack()

Figure 3: Usage Scenario of the Reputation Management System

tion of resource1 regarding this action and updates accordingly the different reputation values User3_* for User3. Some time later, the RCUM service for Resource1 requests to obtain the new reputation value for User3. Based on this new value, RCUM revises its decision to grant access to User3 to use Resource1. This may or may not change the access right for User3. Finally, the VOM decides to end the VO (e.g. as a result of achieving its goals) and informs the RM system of this decision. The RM system acknowledges this decision.

6 ANALYSIS OF THE REPUTATION MODELS

This section describes the results we obtained by performing simulations with various VO setups. We have run our experiments using the SimGrid simulator (Legrand et al., 2003) on which we implemented the following VO operation scenarios:

- VOs with reputation-rated resource providers;
- VOs with reputation-rated users; and
- VOs with reputation-rated resource providers and rated users.

In all simulated scenarios, we compared the results against the case when reputation is not considered to enhance resource management in VOs.

First, we considered a VO with users submitting requests to resource providers for a service. We allow 20% of the providers to produce random QoS values uniformly distributed in a variation band between $85-105\%$ of the agreed SLA expected quality. For scheduling the requests to VO nodes we used the *Res_rep_e* value and we allowed that each node will obtain a number of service reuqests proportionaly with its reputation. For a batch of jobs originating from the VO users, we computed the *total completion time* and the *total welfare* produced in the system. Total welfare is obtained by suming all utilities acquired by the users for the submitted jobs. We varied the load factors of the system. The load factor is defined as the proportion of the requested system capacity at a given moment of time vs the total available capacity to be delivered by the VO. For comparison, we allowed the resource broker to schedule the requests in a round-robin fashion. Figures 4 and 5 show the results.

We should note that with using a reputation-based scheduling, the total completion time is better with around 25% for every load factor of the system. More, the system produses 25% much welfare with reputation.

Next, we simulate the system with unreliable users. An unreliable user tries to execute un-permitted actions. We set the fraction of unreliable users to 20%, each introducing malicious actions with a sabotage rate of 20%. Each action gets a random penalty from $[0,1)$. Each un-permitted request is identified and refused by the system after its execution and its

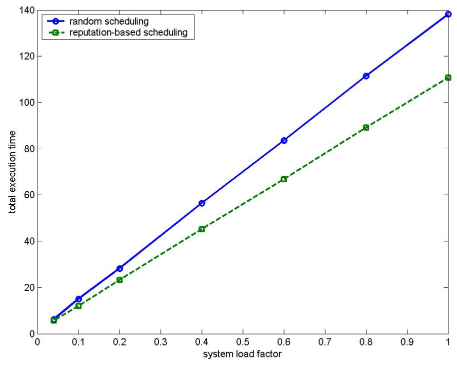

Figure 4: Comparing reputation-based scheduling with round-robin: completion time

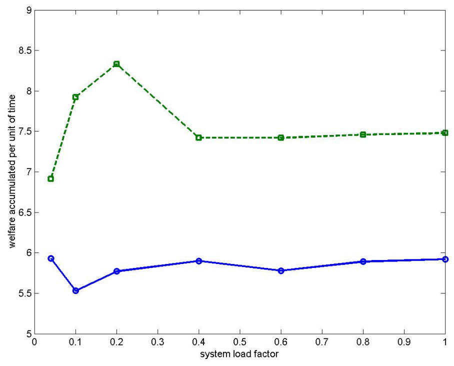

Figure 5: Comparing reputation-based scheduling with round-robin: welfare

result gets discarded and never reaches back the user. To simulate this setup, we use a resource centric brokering approach: for an available resource, the next action/job is selected from the available ones according with the reputation of the user who entered the action in the system. For reputation, we used the *User_rep_e* value.

A more complex scheduling is the one that selects the most reputable available resource and puts on it the job of the most reputable user. This later setup simulates the case of a VO with unreliable users executing actions on unreliable resource providers. Our scheduling intends to protect reputed providers by assigning on them actions from reliable users.

As a benchmark, we used the FIFO (round-robin) scheduling, at each resource executing the oldest request in the system.

Figure 6 shows the simulation results. We counted the total welfare (satisfaction) produced and we depicted the welfare acquisition curve for each of the three cases described above. The *Y* axis plots the proportion of the total satisfaction perceived by the users during the time. We can note that the case when the broker is aware both about the reputation of users and of resources allows for a quicker welfare accumulation. The case when scheduling is done on the basis of first-come first-serve (in both regarding the resources and the user actions) is the worse, letting the users to accumulate satisfaction only latter in time. We should note that by the middle of the simulation, for a total of about 4% ($20\% x 20\%$) malicious actions, we get about 10% more satisfaction acquired.

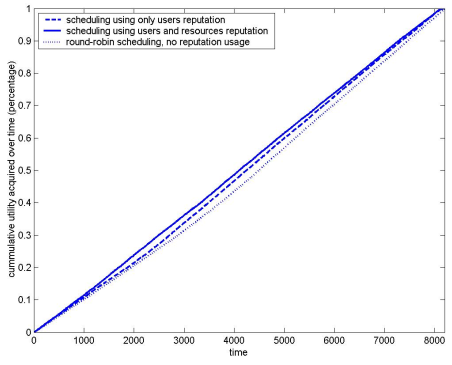

Figure 6: Reputation based scheduling with user and resource reputation. We allowed 20% of users to be malicious and having a sabotage rate of 20%.

We should note that in VOs with reliable users and unreliable resources, the reputation-based approach increases the overall system performance in the sense that tasks get faster executed. With unreliable users, using the reputation-based approach the system increases the user satisfaction by allowing trustfull users to benefit first. Furthermore, the reliable resources are used more effective, in the sense that trusted actions are assigned on them.

7 RELATED WORK

The work presented here builts on the utility-based reputation model developed by Silaghi *et al* in (Silaghi et al., 2007). We have generalised such a model by making explicit generic types such as *Entity*, *Consumer* and *Issue* so that it can be used for maintaining reputation of several type of entities, as shown in section 4. The model has also been simplified by not including the notion of time since similar results can be obtained by interpreting indexes in sequence *Trace* as indicating the time at which an event occurred.

Silaghi's model is based on techniques as those presented in (Huynh et al., 2006; Sabater and Sierra, 2001), which creates reputation from *impressions* collected after a transaction, similar to the events generated by monitors to create reputation. However, in the utility-based model, several reputation events can be generated by each transaction, resulting in a more precise representation of the reputation value.

Our model is close to PathTrust (Kerschbaum et al., 2006), a reputation system for member selection in the formation phase of a VO. When inviting members to join a VO, the initiator selects only those members whose reputation is above a certain threshold and probabilistically selects a member to be in the VO, as we did for reputation-based scheduling in section 6. The reputation is built by aggregating positive and negative feedback the user submits after transaction execution. We consider our model more suitable for Grid systems, since it does not require direct feedback from users, the utility functions measures directly the users' satisfaction.

The GridEigenTrust model (von Laszewski et al., 2005) integrates trust management as part of the QoS management system. They consider both direct and indirect trust, acquired at the level of grid entities and contexts (i.e. service delivery), after transaction execution. It is grounded on the EigenTrust approach (Kamvar et al., 2003), which uses the notion of transitive trust: a peer i has a high opinion of those peers who have provided it good services and therefore, peer i is likely to trust the opinions of those peers. The idea of transitive trust leads to a system where global trust values correspond to the left principal Eigenvector of a matrix of normalized local trust values.

8 CONCLUSIONS

In this paper, we defined a utility-based reputation model geared for Grids. The system is general in the sense that it can be used to rate both users and resources in a VO with respect to issues of interest such as QoS and resource usage. It also allows an entity reputation to be aggregated to the level of the Grid in general.

The model constitutes the basis for a design of a reputation management system that is being currently implemented in EU FP6 project GridTrust. Finally, we carried out simulations of the model to demonstrate its behaviour regarding completion time and welfare, both of which showed improvements over non-reputation-based VOs. We plan to carry out further simulations in order to understand better other behaviour of the model.

ACKNOWLEDGEMENTS

This work is funded by the European Commission under the FP6 IST project GridTrust (project reference number 033817).

REFERENCES

Camarihna-Matos, L. M. and Afsarmanesh, H. (2003). Elements of a ve infrastructure. *Journal of Computers in Industry*, 51(2):139–163.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3).

Huynh, T., Jennings, N. R., and Shadbolt, N. (2006). An Integrated Trust and Reputation Model for Open Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154.

Kamvar, S., Schlosser, M., and Garcia-Molina, H. (2003). The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW '03: 12th International Conference on World Wide Web*, pages 640–651. ACM Press.

Kerschbaum, F., Haller, J., Karabulut, Y., and Robinson, P. (2006). PathTrust: A Trust-based Reputation Service for Virtual Organization Formation. In *iTrust2006: Proceedings of the 4th International Conference on Trust Management*, volume 3986 of *Lecture Notes in Computer Science*, pages 193–205. Springer.

Legrand, A., Marchal, L., and Casanova, H. (2003). Scheduling Distributed Applications: the SimGrid Simulation Framework. In *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, page 138, Washington, DC, USA. IEEE Computer Society.

Sabater, J. and Sierra, C. (2001). REGRET: A Reputation Model for Gregarious Societies. In *Fourth Workshop on Deception, Fraud and Trust in Agent Societies*. ACM Press.

Silaghi, G., Arenas, A., and Silva, L. (2007). A Utility-Based Reputation Model for Service-Oriented Computing. In Priol, T. and Vanneschi, M., editors, *Toward Next Generation Grids*, CoreGRID Series, pages 63–72. Springer.

von Laszewski, G., Alunkal, B., and Veljkovic, I. (2005). Towards Reputable Grids. *Scalable Computing: Practice and Experience*, 6(3):95–106.

Woodcock, J. and Davies, J. (1996). *Using Z: Specification, Refinement, and Proof.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.