SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks **00**: 1–18 (2008)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.0000

Reputation Management in Collaborative Computing Systems

Alvaro E. Arenas*[1], Benjamin Aziz[1] and Gheorghe Cosmin Silaghi[2]

[1]*STFC Rutherford Appleton Laboratory, Oxfordshire, United Kingdom*
[2]*Babeş-Bolyai University, Business Information Systems Department, Cluj-Napoca, Romania*

Summary

In collaborative systems, a set of organisations shares their computing resources, such as compute cycles, storage space, or on-line services, in order to establish Virtual Organisations aimed at achieving common tasks. 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. In this paper, we contribute to research in the area of trust for collaborative computing systems along two directions: First, we provide a survey on the main reputation-based systems that fulfill the trust requirements for collaborative systems, including reputation systems designed for e-commerce, agent-based environments, Peer-to-Peer computing and Grid-based systems. Second, we present a model for reputation management for Grid Virtual Organisations that is based on utility computing and that can be used to rate users according to their resource usage and resources and their providers according to the quality of service they deliver. We also demonstrate, through Grid simulations, how the model can be used in improving completion and welfare in Virtual Organisations.

KEY WORDS: Reputation, Trust Management, Collaborative Systems, Service-Oriented Architectures, Virtual Organisations

1. Introduction

In collaborative systems, a *Virtual Organisation* (VO) can be defined as a set of users and real organisations that provide resources, such as compute cycles, storage space, or on-line services, for users to exploit for a common goal. Examples of common goals include large-scale distributed computing research projects [1] or inter-organisational business applications such as Grid-based supply chains [2], among others. In such collaborative systems, trust management is a fundamental problem as resource owners must share their resources with unknown organisations as well as ensuring that all users abide by the VO agreement to which the resources have been allocated.

This paper investigates how to exploit reputation systems in the management of VOs. Reputation is one measure by which trust among different members of a VO can be quantified and reasoned about (probabilities and insurances being examples of other measures). We focus on collaborative systems where the availability of resources 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

*Correspondence to: Alvaro Arenas, e-Science Department, STFC Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Didcot, Oxfordshire OX11 0QX, United Kingdom.
Email: alvaro.arenas@stfc.ac.uk

Prepared using secauth.cls *[Version: 2008/03/18 v1.00]*

resource providers, according to the Quality of Service (QoS) provided, as well as reputation of users, according to their usage of resources.

Our reputation model is grounded on a utility-based reputation model for service computing [3]. A reputation system gathers, aggregates, and distributes feedbacks about participants' behaviour. The feedback is usually provided as an a-posteriori operation requiring human intervention. This way of building reputation is useful in semi-automated contexts, such as electronic marketplaces where users rate sellers, but it becomes a limitation in fully automated contexts such as Grid-based collaborative systems [4]. In some Grid applications, the resources allocated to a user's job are unknown to the user, and such allocation could change during the job execution, making it difficult to obtain user's feedback. The model introduced in [3] overcomes that limitation by representing users' feedback as utility functions, which takes as input the information provided by trustworthy monitors after a transaction. The utility functions reflect the satisfaction a user perceives after consuming a service; this information is thus used to create reputation about particular resource providers and users.

The main contribution of the paper is to study the impact of applying reputation in 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 provides a survey of the state of the art in reputation systems for Grid and distributed computing. Section 3 describes our model of VOs. Then, Section 4 introduces the reputation model used in the paper, a utility-based model that uses information provided by monitors to rate entities within a Grid. 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 compares our work with others, and finally Section 8 concludes the paper and highlights future work.

2. Background

In this section we will introduce reputation as a mean to assess trust and shortly present the main developments regarding reputation systems for computing environments.

2.1. Reputation: A Means for Assessing Trust

Reputation is a general concept widely used in all aspects of knowledge ranging from humanities, arts and social sciences to digital sciences. It is a concept closely related to trust and it is defined by the Merriam-Webster dictionary [†] as the "overall quality or character as seen or judged by people in general". In fact, reputation is often seen as one measure by which trust or distrust can be built based on good or bad past experiences and observations (direct trust) [5] or based on collected referral information (indirect trust) [6]. In recent years, the concept of reputation has shown itself to be useful in many areas of research in computer science, particularly in the context of distributed and collaborative systems, where interesting issues of trust and security manifest themselves. Therefore, one encounters several definitions, models and systems of reputation in distributed computing research, of which we attempt to touch upon a few briefly in the upcoming sections. For a more detailed survey of reputation models, we direct the reader to [7].

2.2. Reputation Based on Direct Experience

The simplest method to compute the reputation of individuals in open environments is to aggregate the received feedback after transactions and convert it in a reputation measure. This approach is widely used in Internet E-Commerce sites, where users rate the providers.

Internet sites mainly use summation-based centralized reputation systems, based on counting all votes or grades an entity receives. Their big advantage is the simplicity of the reputation scheme. This makes the reputation value to be easily understood by the participants and allows a direct conversion between reputation assessment and trust. The most widely known Internet reputation system of this kind is eBay[‡]. In eBay, after the end of an auction, the buyer and the seller have the opportunity to rate each other's performance with either 1 (positive), 0 (neutral) and -1 (negative). The reputation of a user is the sum on these individual feedback and it is a common knowledge into the system. In eBay, most of the feedback is positive. Although this reputation scheme

[†] http://www.merriam-webster.com/
[‡] http://www.ebay.com

DOI: 10.1002/sec

is very primitive, it works very well, being valitated by its long time existence, acknowledging the Yhprum's Law: systems that shoudn't work, sometimes do, or at least work fairly well.

Similar feedback summation methods were proposed in other e-commerce websites. Beside summation, averaging or weighting the feedback was considered. In the Amazon[§] bookstore, reputation is assigned to books and to reviewers. A reviewer can assign to a book between 1 and 5 stars. The reputation of a book is the average score it received from its reviewers. Reviewers are ranked according to the votes they receive from the users who find their review as valuable. Again, the reputation management system is a centralized one and reputation is a common knowledge in the system. Epinions[¶] reputation system works in a similar way like Amazon.

Zacharia and Maes [8] were among the first ones to build more sophisticated reputation-based systems to overcome existing trust problems in e-commerce online applications. First, they proposed the SPORAS system, based only on direct transaction ratings between users. Users rate each other after a transaction with continuous values from 0.1 to 1. The ratings of a user are aggregated in a recursive fashion, obtaining a reputation value that scales from 0 to 3000 and a reputation deviation to assess the reliability of the reputation value. SPORAS was evaluated to perform better than the eBay and Amazon approaches.

In P2P networks, credit-based reputation mechanisms are employed to assess the individual contribution of the peers. Gnutella-like P2P file sharing systems are among the most popular P2P networks to employ such credit-based schemes. For such networks, [9] proposes a reputation system to track back the past behavior of users and to allow drawing up decisions regarding the content consumers and providers. The reputation is a measure regarding the level of participation of the peers in the system. In this model, the reputation of a peer depends on (1) its behavior assessed in accordance with the contribution of the peer to content search and download and (2) its capability expressed in terms of processing power, bandwidth, storage capability and memory. Each peer in the network gets credit for (1) processing query response messages, (2) serving content and (3) sharing hard-to-find content in the network. Each peer could maintain and compute its reputation locally. But, because there is a misbehavior threat with regard of this operation, a reputation computation agent (RCA) is provided for the P2P network with the goal of keeping track of transactions and of the credits and debits that flows in the network.

P-Grid of [10] is a virtual binary search tree built on a top of a P2P network where reputation is an assessment of the probability that an agent will cheat. The trust is binary; a peer can perform a transaction correctly or not and disseminate only negative information (complaints) as relevant. Each leaf in the virtual tree is associated with a node from the network. A data distribution scheme is envisaged to spread the negative information across the tree. Each node stores also routing information allowing peers to query the system for relevant information in order to locally compute the trust. The decision regarding whether an agent is trustworthy or not is chosen according with the following heuristics: if an observed value for complaints exceeds the general average of the trust measure too much, the peers might be dishonest.

2.3. Reputation Based on Indirect Information

Building trust can be based not only on the past interactions between entities but, also considering the social networks the entity belongs to and the referrals the entity can obtain using the social network. This approach is widely considered in multi-agent research, where [11] defines the concepts of agent communities and social networks. The members of an online community provide services and referrals for services to each other. A participant in a social network has reputation for both *expertise* (providing good services) and *sociability* (providing good referrals). This theoretical framework is conceptually valid also for P2P service-oriented systems. Items under study concern the way the reputation is represented, how referral information is aggregated, which learning model is used.

2.3.1. Reputation in Multi-agent Systems

In general, in agent-based systems, agents register locally the direct and indirect trust values at various time moments and use them in aggregation formulas to derive the reputation measures.

Abdul-Rahman and Hailes [6] propose a model with discrete values of trust: very trustworthy, trustworthy, untrustworthy and very untrustworthy. Each agent builds its local reputation, storing both the direct and indirect trust values. Singh et al. [11] further

[§]http://www.amazon.com
[¶]http://www.epinions.com

DOI: 10.1002/sec

refines the referral model of [6], using the vector space model for representing the local trust values. After each transaction, an agent updates its partner expertise after verifying the delivered QoS. If the QoS is bad, therefore, for the whole chain of agents who referred that partner the sociability measure is decreased.

HISTOS [8] is a follower of SPORAS, taking into account also the social network created between users through performing transactions. The reputation model for user A_0 from user A_i point of view considers all paths on the social network graph between these two users. The same kind of social network was used after that in the approaches of [12, 13].

In [14], a model, named REGRET, is proposed that considers the following dimensions of the reputation: *the individual dimension*: which is the direct trust obtained by previous experience with another agent, *the social dimension* which refers to the trust of an agent in relation with a group and *the ontological dimension* which reflects the subjective particularities of an individual. Their model focuses on SLAs between two parties, with several variables under interest. This model is easy to be understood and allows one to express the reputation for the individual and the group-related experience and to compose services by aggregation.

In [15], the REGRET system is further refined by detailing more on the terms of a contract (Service Level Agreement - SLA), with the intention to build a trust model for the negotiation of a contract. Agents get some utility after the execution of a contract and each partner in a contract should have some expectations about the outcome values of the issues. Agents behavior is restricted by some general rules of the society they belong to, the rules of the restricted group of the agent and the institutional rules that originate from the contracts the agent is involved in. The trust model is composed by two components: *confidence* - accounting for the direct trust (obtained only by the agent's experience) and *reputation* - accounting from the trust obtained from the social environment.

Building upon the previous two reputation models, [16] propose a trust model with the following four dimensions: (1) interaction trust, (2) role-based trust, (3) witness reputation and (4) certified reputation. Interaction trust comes for the direct experience. Ratings are aggregated using a sophisticated time-discount function, putting more emphasize on newer information. Role-based trust results from the role-based relationships between two agents (e.g. owned by the same company, the relationship between a service provider and its users etc). The witness reputation is obtained from the social network of the agent, following a query-based referral process. The certified reputation of an agent consists of a number of certified references about its behaviour on a particular task. They showed that each component of the model adds an improvement in how reliable and fast an agent finds its partners in transactions.

2.3.2. Belief-oriented Trust

These models keep valid the basic assumptions of the referral networks. They refine the above described models by considering the fact that trust is a human *belief* involving a subject and an object and the trust in a system is a *subjective* measure. Because of the imperfect knowledge about the reality, one might only have an opinion about trusting an object and this opinion could be a belief, disbelief and uncertainty [17]. The roots of this approach are in the Dempster-Shafer theory of evidence and this approach is consistent with the theory of Marsh [18], allowing the existence of two thresholds for expressing trust and untrust beliefs. Trust values are continuous in this case and the storage model can be distributed at the levels of the nodes in the system.

Jøsang [17, 19] derives the Beta reputation model, based on the subjective logic. The Jøsang's subjective logic is a trivalent one, an opinion having three degrees of positive values: belief (b), disbelief (d) and uncertainty (u), with $b + d + u = 1$. b, d and u can be assessed from the previous experience of the agent with the object of the trust using the beta distribution function, which is applicable in a space where every event can be successful or unsuccessful. Each agent can apply the following operators to produce its internal trust model: (1) the conjunction operator in order to infer a conclusion about a proposition, having two opinions about that proposition (2) the consensus operator between independent and dependent opinions (3) the recommendation operator, allowing the agent to include in the inference chain the recommendations received from a referral.

In [20], [21] and [22], reputation systems are developed based on the Beta model described above. Furthermore, the model of Yu and Singh [23] is more expressive than that of Jøsang's as they allow continuous values in order to assess the outcome of a transaction. On their model, they directly use the Dempster's rule of combination in order to aggregate two belief functions built on different evidences. This operator has the same meaning as the conjunction

DOI: 10.1002/sec

operator in the Jøsang's model. In [23], an agent local decision model is described for selection of a transaction partner based on beliefs.

2.3.3. P2P Approaches

In P2P systems, one main concern is the identification of malicious peers that provides misleading services. Trust models might prevent such behavior and might improve the reliability and fault tolerance of the system. In a P2P approach, the challenge is how to aggregate the local trust values without a centralized storage management and facility. Beside, two kinds of questions are addressed by P2P approaches: what trust metric should be considered and how to store reliably and securely the trust values across the network. P2P reputation systems should be [24]: (1) self-policing: no central authority should exist and the peers should enforce the ethical behavior by themselves, (2) anonymous: peer reputation should be associated with an opaque identifier, (3) the system should not assign profit to newcomers, (4) minimal overhead and (5) robust to malicious collectives of peers.

Rather than the approach of [9] which is a transaction-based one in a Gnutella-like P2P file sharing system, TrustMe [25] is a user-based approach, adopting the principle of obtaining references about a peer, before engaging in a transaction with that peer. Broadly, TrustMe functions in the following manner: each peer is equipped with a couple of public-private key pairs. Trust values of a peer B are randomly stored at another peer *THA* in the network. Any peer A interested in the trust value of a peer B broadcast the query on the network and the *THA* peers replies this query. After interaction, peer A files a report for peer B indicating the new trust value for B and therefore, *THA* can modify the trust value of B accordingly. TrustMe uses a smart public key cryptography mechanism to provide security, reliability and accountability.

PeerTrust [26] is based on five important parameters contributing to a general trust metric: (1) the feedback a peer obtains from other peers, (2) the feedback scope counted as the number of total transactions that a peer has with other peers, (3) the credibility factor for the feedback source, (4) the transaction context factor discriminating between mission-critical and non-critical transactions and (5) the community context factor for addressing community-related characteristics. Credibility is assessed either using recursively the local trust values, either using the similarity between satisfaction vectors collected from the peers. This model resembles with the models from the agent world [14, 15, 16] where reputation is assessed on various important dimensions.

A referral network is used in [12] as a source for obtaining recommendations in a P2P environment. Like in P-Grid, trust is a probabilistic measure. A graph overlay is designed on the top of the P2P network by linking peers who have direct transactional relationship or who are indirectly socially linked by referrals. In their approach, they model the ability of a peer to make recommendations, which is different from the peer trustworthiness. Assuming a probability l_k that a peer p_k lies, one can derive the probability of observing a good or bad report from peer k about peer j. Given a sample of independent reports about peer j, one can compute the likelihood behavior of j, which in turn, depends on the internal probability of agent j for performing honestly. Maximizing this likelihood, one can obtain the probability associated with a peer. The model is further detailed by considering several services provided by the peers.

The NICE approach [13] targets a specific P2P network implementation, the NICE platform. The trust value of a node B at a node A is a measure of how likely the node A believes a transaction with node B will be successful. In NICE, each peer is equipped with a pair of public and private keys and the messages are signed by their creators. After each transaction between a client A and a servant B, A generates a cookie with its perceived feedback (trust value) for the transaction and this cookie can be stored by the servant B. B can decide which cookies to store and how long to store such a cookie. More, each peer could posses its own algorithm for updating and storing the trust values it receives from transaction partners. When a peer A deliberates to enter a transaction with peer B and A has no direct evidence (cookies) for B, A will ask its partners about having cookies for B and the partners will continue to spread the request into the network till a path between A and B is established, building the social network graph. The paths between A and B are evaluated either by selecting the minimum trust value on the path or by multiplying the trust values. Therefore, the strongest path can be selected. When testing this protocol against malicious peers (peers that do not follow the NICE trust protocol), the authors observed a robust cooperative emerging in the system.

EigenTrust [24] proposes a robust distributed method to compute the global trust in a fully decentralized P2P network, where each peer stores locally the trust values for all the rest of peers in the system. Their approach is based on the notion

DOI: 10.1002/sec

of transitive trust: a peer i has a high opinion of those peers who have supplied with 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. At every iteration of the EigenTrust scheme, each peer will interrogate its neighbors for their local trust values and further refine and distribute the global trust vector partially computed.

2.4. Incentive Compatible Approaches

In its classical understanding, reputation is an supplemental tool in order to enhance other critical mechanisms inside the computing environment, such us the resource management. Usually, agents are assumed to report truthfully either the direct feedback or their opinion about others. But, as observed by [27], it is not in the best of an agent to (i) report reputation information because it provides a competitive advantage to others; (ii) report positive ratings because the agent slightly decrease its own reputation with respect to the average of other agents and therefore, reporting negative ratings the agent will increase its own reputation. Therefore, reputation might be combined with an incentive mechanism in order that reputation sharing to become a rational alternative for an agent. The game-theoretical approach to reputation directs the research in this direction. Side-payments are envisaged in [27] as incentives for agents. Some R-Agents manages the reputation and the side-payments in the system. Each agent that needs reputation information for a partner will buy this information from the R-Agents. Each transaction in the system is monetary valued. After the transaction, each agent can submit paid reputation reports to the R-Agents. R-Agents will pay positive amounts for similar reports regarding a target agent. They showed rational behavior is mutually enforced if the environment consists in majority of honest agents.

The authors extended the model for pricing services in P2P networks [28] and for improving the service level agreement in the web services world [29].

2.5. Reputation Management in Grids

Reputation models can bring with more dependability in the grid either by tackling the sabotage tolerance issue, improving resource allocation and scheduling or fine-graining the user access. In all cases, the usage of reputation models affects the notion of trust in the grid computing environment, allowing the system to construct the *soft* version of trust.

Sabotage tolerance problem is specific for desktop grids [30] that collect idle computing power from desktop computers gathered voluntarily from Internet. Workers that participate in the system have incentives to sabotage the computation in order to get more credit in the system. Credibility-based fault tolerance is introduced in [31] that combines result verification by task replication with a reputation mechanism. Each worker scores has a credibility related with his past behavior. The credibility of a worker is further transfered to his submitted results and to the group of results related with a work entry. The credibility of a result group is an estimate of the conditional probability that the result being correct. A result will be accepted if it counts at least some minimum threshold v.

A trust-based scheduling model is defined in [32] in desktop grids based on the principle of ”trust and verify”. Each worker will be verified by spot-checking and will gather reputation by successfully passing the verification. Workers with higher reputation will require less verification than ones with a lower trust values. In [33], the reputation mechanism is combined with a weighted voting scheme to decide about the requested level of replication necessary to accept a result.

In classical grids, Laszewski et al. [34] exploit the beneficial properties of EigenTrust [24]. They integrate the trust management system as part of the QoS management framework, proposing to probabilistically pre-select the resources based on their likelihood to deliver the requested capability and capacity. Considering two organizations whose entities interact, a trust table stores the direct trust between the organizations, for each context of the transactions. The global trust between organizations at time t is computed by weighting the direct trust table entry with a time decay weight. The trust relationship of organization i for another organization j for a context c is obtained by aggregating the direct trust between these two organizations with the reputation of organization j, weighted with normalized values. The global trust or reputation of an organization j is computed by obtaining recommendations from a third organization and by aggregating the received recommendations with the direct trust values, applying the time decay function specific for the given context. This value is normalized as to scale to $[0, 1]$.

PathTrust [35] is a reputation system proposed for member selection in the formation phase of a virtual

DOI: 10.1002/sec

organization. To enter the VO formation process, a member must register with an Enterprise Network (EN) infrastructure by presenting some credentials. Besides user management, EN supplies with a centralized reputation service. At the dissolution of the VO, each member leaves feedback ratings to the reputation server for other members with whom they experienced transactions. The system requires each transaction to be rated by the participants. PathTrust arranges the participants in graph structure similar with the social network described in the agent approaches. Each edge in the graph is weighted with the trust between the nodes at the ends of the edge, computed by counting all the positive feedback between the nodes and subtracting all the negative one. Like NICE [13], the graph is used to assess the reputation between two nodes that do not have direct transactions.

In this paper we will use a reputation approach devised for service oriented computing, inspired from the benefits of the above-presented models and try to overcome the main limitation of the classical feedback-based reputation models: the subjectivity of the opinions. As we have seen in this section, all reputation models based on posteriori-collected feedback should consider that the content of these messages might be intentionally distorted by malicious reporters.

3. A Model of Virtual Organisations

In order to support rapid formation of VOs, we use the concept of *virtual breeding environment* (VBE) [36] adopted from the Virtual Enterprises community. 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 services, represented by ovals, and include users that utilise VO services, represented by small squares. Organisations pre-register to a VBE via the VO Manager component, including description of the services they are willing to share in a Grid and the list of potential users belonging to the organisation. When a user wants to create a VO, he assumes the role of VO Owner and contacts the VO Manager with the description of the needed services. The VO Manager includes a service brokering component that suggests potential concrete

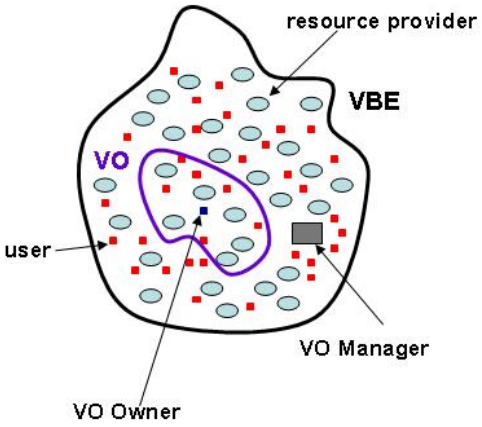

Fig. 1. VBE and VO Models

services and their service providers. The VO Owner then selects a subset of these service providers and their services, and then defines the list of users for the VO.

The VBE can be seen as a market place where service providers are competing to participate in VOs and users within VOs are competing to use services. Reputation information about service providers can be used as a parameter for guiding the selection of VO partners. On the other hand, having reputation information about users could help service providers to implant tighter security mechanisms for accessing their services and the resources underlying them.

4. A Utility-Based Reputation Model for VOs

In this section, we develop a general utility-based reputation model for VOs, which will be used later to manage reputation in service-oriented VOs. Our reputation model is based on the model described in [3]. The model was initially devised for service-oriented computing in grid systems and improves the models presented in subsection 2.5.

Central to our model is the notion of an *organisation*. The set of all organisations is denoted by *Org*. 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 for 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 *consume* (use) the entities

DOI: 10.1002/sec

and qualify them are members of the set *Cons*. The sets *Ent*, *Issue* and *Cons* are considered type parameters that will be instantiated 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 [37].

$$[Org,\ VOId]$$

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

$$[Cons, Ent, Issue]$$
$$EntOrg : Ent \fun Org$$
$$EntIssue : Ent \fun \mathbb{P}\, Issue$$
$$EntCons : Ent \pfun \mathbb{P}\, Cons$$

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

In our model, we assume 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, at an observed moment of time. We represent an event as a tuple that contains of the following elements: the *time stamp* of the event, a *consumer*, an *entity*, an *issue*, *VOId* and a k number of attributes, *Attr*. A trace corresponds to a sequence of events:

$$\begin{aligned} Event &== TimeStamp \times Cons \times Ent \times Issue \times \\ &\quad VOId \times Attr_1 \times \ldots \times Attr_k \\ Trace &== \mathrm{seq}\ Event \end{aligned}$$

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

$$utility : Event \fun [0,1]$$

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.

$$[Cons, Ent, Issue]$$
$$rep\_eic : Time \times Cons \times Ent \times Issue \times VOId \fun [0,1]$$
$$\forall\, c : Cons; e : Ent; i : Issue; vo : VOId \bullet rep\_eic(t,c,e,i,vo) = \frac{\sum_{ev \in Trace \upharpoonright \{(ts,c,e,i,vo,\cdots) \in Event\}} \varphi(t,ts)\, utility(ev)}{\#(Trace \upharpoonright \{(ts,c,e,i,vo,\cdots) \in Event\})}$$

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. $\varphi(t,ts)$ is a time discount function that puts more importance on events registered closer in time with the moment of computing the reputation. Reputation, *rep_eic*, is defined as the weighted 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*.

As a fitness measure for the above-defined reputation, we consider the reputation deviation *dev_rep_eic*. The reputation deviation shows how much the reputation varies in time and it evaluates the stability in the behavior of the VO members.

$$[Cons, Ent, Issue]$$
$$dev\_rep\_eic : Time \times Cons \times Ent \times Issue \times VOId \fun [0,1]$$
$$\forall\, c : Cons; e : Ent; i : Issue; vo : VOId \bullet dev\_rep\_eic(t,c,e,i,vo) = \frac{\sum_{ev \in Trace \upharpoonright \{(ts,c,e,i,vo,\ldots) \in Event\}} \varphi(t,ts) |utility(ev) - rep\_eic(t,c,e,i,vo)|}{\#(Trace \upharpoonright \{(ts,c,e,i,vo,\ldots) \in Event\})}$$

Aggregating the reputation of an entity over all its consumers within a VO produces the reputation of the entity in the VO with respect to a particular issue of interest.

$$[Ent, Issue]$$
$$rep\_ei : Time \times Ent \times Issue \times VOId \fun [0,1]$$
$$\forall\, e : Ent; i : Issue; vo : VOId \bullet rep\_ei(t,e,i,vo) = \frac{\sum_{c \in EntCons(e)} rep\_eic(t,c,e,i,vo)}{\#EntCons(e)}$$

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 its reputation in each of its issues of interest within that VO.

DOI: 10.1002/sec

$$[Ent]$$
$$rep\_e : Time \times Ent \times VOId \rightarrow [0,1]$$
$$\forall e : Ent, vo : VOId \bullet$$
$$rep\_e(t,e,vo) = \frac{\sum_{i \in EntIssue(e)} rep\_ei(t,e,i,vo)}{\#EntIssue(e)}$$

The general VBE reputation of an entity is then the aggregation of its reputation in all VOs.

$$[Ent]$$
$$rep : Time \times Ent \rightarrow [0,1]$$
$$\forall e : Ent \bullet rep(t,e) = \frac{\sum_{vo \in \mathrm{dom}\ VOS} rep\_e(t,e,vo)}{\#\,\mathrm{dom}\ VOS}$$

This last definition assumes that the domain of *VOS* represents the VBE; i.e. all the organisations participating in it.

Reputation deviation can be defined for all above-presented reputation deviation.

4.1. Properties of the Reputation Model

In this section we shortly discuss the properties of the reputation model presented above.

At the beginning we should note the usage of monitors to gather data for building the reputation. Most reputation models presented in section 2 are based on direct or indirect feedback collected from information sources with questionable reputation. As many VBEs (like Grids) supply with trustable monitoring services, using the data provided by those services seems to be a good alternative.

Next, central to the described reputation model resides the utility functions. Utility functions reflect the consumer perception about the delivered services. The utility function is an intrinsic evaluation of each consumer and might be difficult to construct it. We assume that somehow, the reputation manager builds the utility function of each consumer *before* the service to be delivered. For example, every user that joins a VBE and registers to the reputation management service can pass a questionnaire used to further infer the utility function for that user. Knowing the utility functions before service delivery is a key feature of our model because it reduces the risk of cheating. If the consumer reveals out another utility function than its real one, it will end up with a service delivery associated with this wrong utility function, thus, will not benefit any more from the participation in the collaborative system.

When computing the reputation, instantaneous utility values associated with the events are weighted using a time discount function. Putting more emphasize on newer events represents a widely accepted approach in the reputation management literature [14, 15, 16]. We recommend the following time discount function, also adopted by [16]:

$\varphi(t,ts) = e^{-\frac{t-ts}{\lambda}}$

λ is a parameter used to tune the importance of the newer events against older ones and its value is related with the time scale employed by the monitors for registering the events.

Figure 2 depicts the reputation for a service with one issue uniformly delivered in a variation band of 85% to 105% of the agreed QoS for the issue. On the top plot we depicted the cloud of the observed events. We considered 1000 time units. We can notice that at the begining of the experiment, after a short learning time frame, the reputation stabilizes itself around a value.

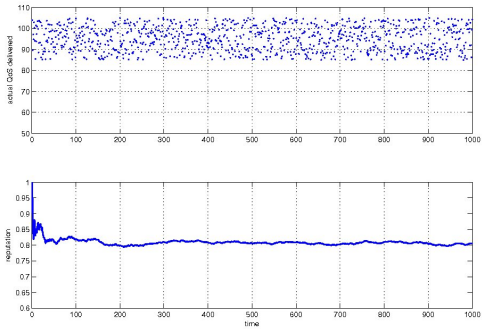

Fig. 2. Reputation when the issue is delivered uniformly distributed in a variation band

Figure 3 we considered a similar setup, but now, between time units 200 and 300 the consumer perceived a decay in the QoS. We can note the reputation recovers slowly after the dramatic decay in the QoS and on the long time, as the consumer gets continuously the same delivery patters as before the decay, the reputation converges to the initial value.

Reputation deviation can be used to decide in cases the service is delivered with a fluctuating quality. A reputation value accompanied with a smaller reputation deviation indicates a higher confidence in the expected value for the item under study. For example, normal distributed values around a central average are less fluctuating than uniformly distributed values around the same average. The upper part of figure 4 shows the how reputation varies in time for two patterns of QoS delivery explained above. In

DOI: 10.1002/sec

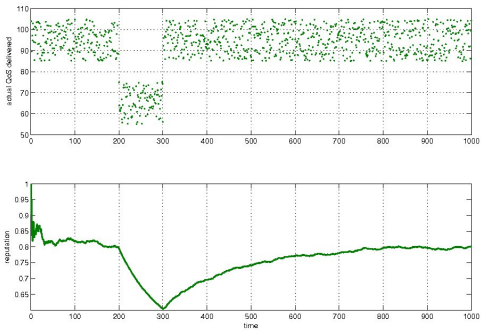

Fig. 3. Reputation when there is a decay in the QoS delivery

the lower part of the figure we can notice that the reputation deviation curve for the case of the normal delivery is situated below the curve for the uniform delivery of the QoS, indicating much confidence on the reputation of the first provider.

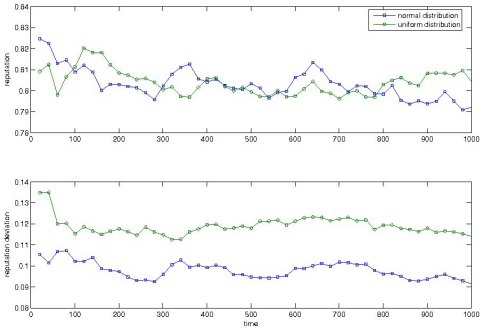

Fig. 4. Reputation deviation used to assess the confidence in the reputation value

4.2. Reputation Management for VO Service Providers

Here we aim at maintaining reputation for organisations as service providers in a VO according to the Quality of Service (QoS) of the services they provide. In this model, *consumers* correspond to the *users* in a VO, denoted by *VOUser*, and *entities* correspond to the *VO services*, denoted by the set *Srv*. There are several options for selecting issues of interest. We can have either a fine granularity where each service level objective defined for a service can be seen as an issue of interest, or a coarse granularity where the whole QoS can be seen as a single issue of interest. For simplicity, we select the latter option.

Functions *UsersVO* and *SrvVO* represent the set of users of a VO and the services that an organisation provides to a VO, respectively.

$$UsersVO : VOId \rightarrow \mathbb{P}\, VOUser$$
$$SrvVO : VOId \times Org \rightarrow \mathbb{P}\, Srv$$

As we mentioned earlier, our model requires the existence of monitors capable of detecting variations in the QoS of each service and generating events to inform the reputation system about such variations. An event is then represented as a tuple denoting the current value of the QoS of a service being used by a user within a VO.

$$Event == TimeStamp \times VOUser \times Srv \times \{QoS\} \times VOId \times \mathbb{R}$$

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 service provider for a particular service within a VO.

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

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

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

We can now define the reputation of a service using the reputation functions defined in the previous section. Here *Srv_rep_eic* denotes the reputation value given by a particular VO user to a service in relation to its QoS in the VO. *Srv_rep_ei* represents the reputation of a service taking into account its QoS in a VO; it is an aggregation of the reputation given by all users to the service in relation to the QoS issue of interest within that VO. *Srv_rep_e* denotes the reputation of a service in a VO. Finally, *Srv_rep* indicates the general reputation of a service. Note, since we have only one issue of interest, *Srv_rep_ei* and *Srv_rep_e* will be equivalent. All these reputation values are computed at a given time moment.

DOI: 10.1002/sec

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

Using the above functions, we can now define the reputation of an organisation in a VO and its reputation in a VBE. The reputation of an organisation in a VO, denoted by *Org_rep_VO*, is defined as the aggregation of the reputation of all the services it provides to that VO.

$$
\begin{array}{|l}
Org\_rep\_VO : TimeStamp \times Org \\
\quad \times VOId \rightarrow \mathbb{R} \\
\hline
\forall\, o \in Org, vo \in VOId \bullet \\
Org\_rep\_VO(t, o, vo) = \\
\qquad \dfrac{\sum_{e \in SrvVO(vo,o)} Srv\_rep\_e(t,e,vo)}{\#SrvVO(vo,o)}
\end{array}
$$

On the other hand, the general VBE reputation of an organisation is defined as the aggregation over all its VO reputations.

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

4.3. Reputation Management for VO Users

Next, we discuss how to maintain the reputation of users within a VO and within a VBE according to their usage of VO services. In this model, *users*, denoted by set *VOUser*, are seen as *entities* who could execute some pre-defined actions on services following pre-established policies. *Services*, denoted by set *Srv*, are seen as *consumers* that qualify users in relation to their actions. If a user attempts to execute an action that is not allowed by the VO policy, it will be given a bad qualification by the service that would be reflected in the user's reputation. In some sense, the model here reverses the notions of consumers and entities with respect to the model of previous section.

The set, *Action*, denotes the set of possible actions that can be performed on services. A policy indicates the set of actions a user is allowed to perform on the service in a VO. It represents the expected behaviour of the user.

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

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

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

Now, events are defined as follows:

$$
\begin{array}{lcl}
Event & == & TimeStamp \times Srv \times VOUser \times \\
 & & \{Usage\} \times VOId \times Action
\end{array}
$$

where *Usage* is a name indicating the service-usage issue of interest. We assume the existence of functions, *policy* and *penalty*, that are used to define the utility that a service gets according to the actions performed by a user in a VO. These functions are domain-specific, and based on them, once can define the utility function as follows.

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

We can now define the reputation of a user using the reputation functions defined in Section 4. Here *User_rep_eic* denotes the reputation value given by a particular service to a VO user in relation to the *Usage* of the service in a VO. *User_rep_ei* represents the reputation of a user taking into account its service usage in a VO; it aggregates the reputation of the user for all services 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 issues 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.

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

5. A Reputation Management System

Here we present the architecture of a VO that uses reputation management in order to facilitate the rating

DOI: 10.1002/sec

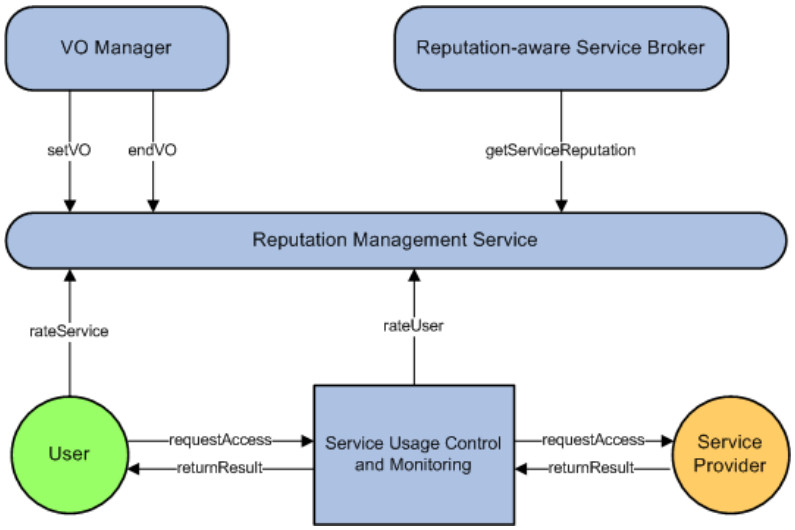

Fig. 5. Reputation Management in VOs

of both VO services and users. The architecture was implemented in the EU FP6 project GridTrust[‖]. 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 Service Broker* (RSB) and a *Service Usage Control and Monitoring* (SUCM) service. The system is shown in Figure 5.

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

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

Where *VO ID* is the identity of the VO being registered, *Service ID List* is the list of services 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 RSB service is used during the setting of new VOs by the VOM. During this phase, the RSB may request from the RM service the reputation of a service in a particular VO or in the general VBE before proposing it to the VOM:

RSB→ RM : getServiceRep(Service ID, VO ID)
RM→ RSB : return(Service ID, Reputation Value)

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

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

SUCM→ RM : reportUser(Service ID, User ID, VO ID, Action)
RM→ SUCM : ack()

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

User→ RM : rateService(User ID, Service ID, VO ID, QoS Value)
RM→ User : ack()

[‖] http://www.gridtrust.eu

DOI: 10.1002/sec

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 6. We assume that a RSB starts by querying the RM system for the reputation of a couple of services, Service1 and Service2, 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 services 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 services offered.

At some stage, the SUCM service at Service1 captures a prohibited action performed by User3 and thus reports it to the RM system. Based on the utility function for Service1 and the penalty for the prohibited action, the RM system computes the satisfaction of Service1 regarding this action and updates accordingly the different reputation values for User3. Some time later, the SUCM service for Service1 requests to obtain the new reputation value for User3. Based on this new value, SUCM revises its decision to grant access to User3 to use Service1. 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 [38] 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.

We emphasize from the very beginning on the advantages of our reputation model. First, because we base our model on a-priori collected utility functions, we eliminate one of the biggest drawback of reputation-based trust: the subjectivity of the collected feedback. If the service consumer does not provide truthful information, the provider will be unable to compile the real utility function and will deliver the service accordingly. Thus, as the rational consumers desire to obtain the expected quality of service, they can not manipulate the reputation of a provider without suffering themselves from worsening the received service quality.

Second, our virtual environment is equipped with a trustful reputation manager. Thus, all messages containing valuable information for computing the reputation can not be tampered by malicious nodes. As the reputation manager collects all the individual utility functions, it can unify the perception about the service delivery via the computed reputation measure.

With our reputation model we tackle a major drawback of the quality of service measurement: the fact that various nodes in the system have different representations about which is a good quality delivery.

6.1. VOs with reputation-rated resource providers

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 7 and 8 show the results.

We should note that with using a reputation-based scheduling, the total completion time is better with

DOI: 10.1002/sec

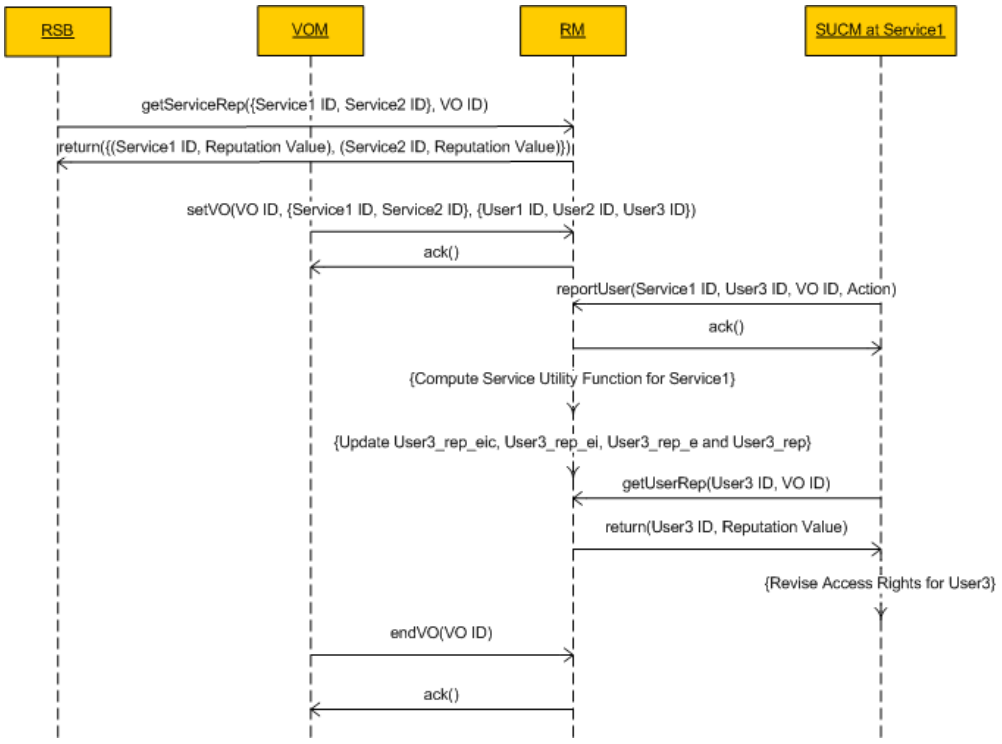

Fig. 6. Usage Scenario of the Reputation Management System

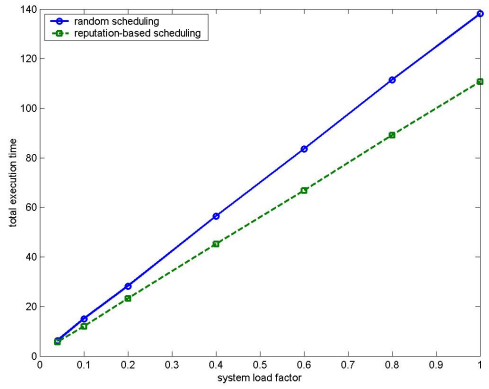

Fig. 7. Comparing reputation-based scheduling with round-robin: completion time

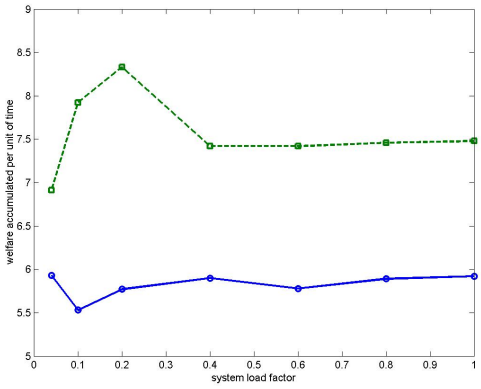

Fig. 8. Comparing reputation-based scheduling with round-robin: welfare

around 25% for every load factor of the system. More, the system produses 25% much welfare with reputation. The gain in welfare was registered due the fact that the reputation manager has a global view about the quality of the service - through the a-priori collected utility functions. Thus, it can enhance the borkering by selecting appropiate providers for a given consumer.

6.2. VOs with reputation-rated users

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 $f = 20\%$, each introducing malicious actions with a sabotage rate of $s = 20\%$. 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

DOI: 10.1002/sec

result gets discarded and never reaches back the user. We should note that the system is now loaded with a fraction *fs* of malicious requests and it spends some useless time to fulfill those requests.

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. Such a broker will postpone the execution of an action originating at a less reputed user as late as possible, maximizing the efficiency of the spent running time of the resource. Figure 9 shows how the proportion of time spent on processing trustful actions varies during the VO existence. We compared the reputation-enhanced broker with a broker that schedules the tasks on the first-come first-served principle.

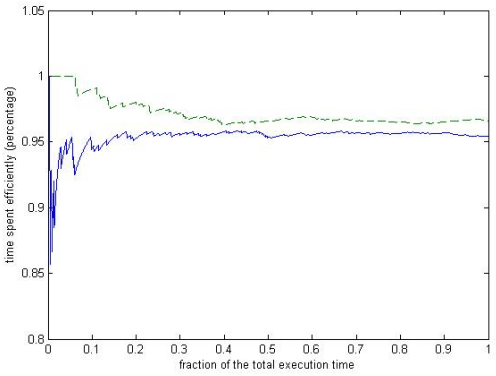

Fig. 9. Time consumption efficiency for a system with malicious users. We allowed 20% of users to be malicious and having a sabotage rate of 20%.

We can note that at the beginning of the VO life, the reputation-enhanced broker (depicted with dotted line) schedules only tasks/actions coming from reputed users. The broker starts to schedule tasks from less reputed users only when tasks get scarce in the VO. More, we can note that the overall fraction of time spent on tasks computing is bigger than in the case of a first-come first-served broker. Thus, the reputation manager succeeds to perform a distinction between malicious users and the truthful ones, enabling an enhanced brokering of the reputed users tasks.

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

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 10 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%x20%) malicious actions, we get about 10% more satisfaction acquired.

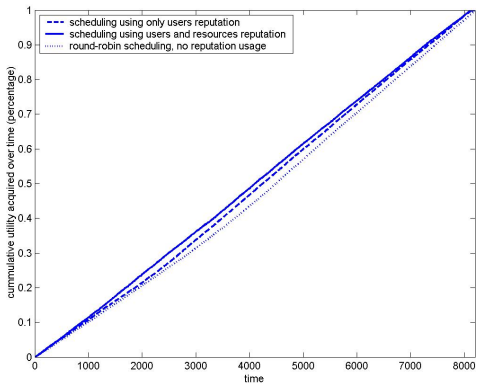

Fig. 10. Reputation based scheduling with user and resource reputation. We allowed 20% of users to be malicious and having a sabotage rate of 20%.

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.

DOI: 10.1002/sec

7. Discussion

In this section we compare our reputation model with the main related models reviewed in Section 2 according to the following criteria: uniqueness of the identities of the participating entities, searching reputation information, reliability of reputation information, and contextual management. These criteria have an impact on managing reputation for dynamic VOs.

- *Unique identities*.

Identity uniqueness is concerned with the question of whether there is a global principal identification mechanism or not. In the case of our reputation management system, both users and services are identified uniquely (Users with an X509 certificate and service with a unique id) within the scope of a single VBE. Our model in this respect is similar to the NICE approach of [13], which also considers unique identities but does not focus on the issue of how trust-based data will be associated with such identities. However, our model is limited with respect to providing inter-VBE unique identification.

- *Searching reputation information*.

This criterion is concerned with the manner in which an entity finds trustworthiness information about other entities and the scope of that information. Usually, this is either done in a direct or an indirect manner and the scope of the information could range from the individual to the community-wide. In our model, the reputation values of service providers and their services or users are obtained directly from the Reputation Management service and reputation is not obtained indirectly by passing through middle entities. This is unlike P2P solutions (Section 2.3.3), where each entity maintains a list of its recommenders or neighbours, and thus when the entity needs the required trust data about another entity, it contacts its neighbours. If a neighbour has the desired data it returns the data to the entity, otherwise it contacts its own neighbours and so on, therefore creating a chain of trust. The process continues until either an entity with the desired data is found, or the chain limit is reached.

The scope of the reputation in our model could be either individual (*rep_eic*) or community-wide as represented by the VBE reputation (*rep*). In the latter case, the VBE is seen as the community itself and our model maintains reputation within its scope. This is similar to the solution adopted in PathTrust [35], where an enterprise network (a concept similar to VBE) is seen as the community. In other systems (for example, REGRET [14]), small groups of communities are formed; in this approach an entity's own experience is checked and the group is consulted for opinions.

- *Reliability of reputation information*.

The question of the reliability of the information used in calculating the reputation of entities is an important one as it affects trust-based decisions. There are several approaches to determine the reliability of reputation information. For instance, P-Grid [10] assumes that if an entity is trustworthy, then it is also reliable in providing recommendations. Other systems separate the two cases: the general reputation and the specific one in providing recommendations (for instance [23]).

An interesting approach is the use of weights as a measure of the reliability of trustworthiness information as was adopted by [19] in the Beta Reputation system. In our model, such weights can be added in as an attribute to the events generated by the reputation monitor and the definitions of the various reputation values can be adjusted accordingly.

- *Context Management*.

Reputation depends on a context. In our model, context is represented by the issue of interest variable, allowing us to keep reputation on several contexts (issues of interest) simultaneously. In many existing reputation solutions, there is a very limited or no consideration of the context in which reputation is mentioned (for example, the EigenTrust algorithm [24]). PeerTrust [26] provides a coarse mission critical and non-critical classification of the contexts of transactions. On the other hand, Laszewski et al. [34] allow contexts to be considered in a limited manner.

8. Conclusion and Future Work

Trust in distributed computing is an important area of research, which contributes to the strengthening of security and the enhancement of the robustness of information sharing. Reputation is one measure by which trust can be quantified and reasoned upon.

Our main contribution in this paper was to define a model for reputation management in service-oriented virtual organisations that is based on utility computing and that can be used to rate users according to their

DOI: 10.1002/sec

service usage and service providers and their services according to the quality of service they deliver. We also demonstrated, through Grid simulations, the behaviour of the model regarding completion time and welfare, both of which showed improvements over non-reputation-based VOs.

There are several areas in which our model can be extended and improved. The portability of reputation across VBEs is one such area in which an entity or a consumer who change their community (VBE) can bring along their reputation from previous communities to any new ones they join. Another area is to enhance the model to be able to express the reliability of events by adding weights to them. Also, the language of the issues of interest is currently left undefined (apart from the service QoS and service Usage terms). Such a language is domain-specific and it can be obtained from domain-specific ontologies. Finally, an interesting area is to model the reputation of orchestrated services based on the reputation of the individual services. This requires the definition of a *reputation composition* operator.

Acknowledgements

The authors are grateful to the constructive comments and useful feedback they received from all the anonymous reviewers of this paper. Gheorghe Cosmin Silaghi acknowledges support from the Romanian Authority for Scientific Research under contract IDEI_2452.

References

1. Britton D, Cass A, Clarke P, Coles J, Colling D, Doyle A, Geddes N, Gordon J, Jones R, Kelsey D, *et al.*. GridPP – The UK Grid for Particle Physics. *7th UK e-Science All Hands Meeting*, 2008.
2. Blasi L, Arenas A, Aziz B, Mori P, Rovati U, Crispo B, Martinelli F, Massonet P. A Secure Environment for Grid-Based Supply Chains. *eChallenges 2008 Conference, Collaboration and the Knowledge Economy: Issues, Applications, Case Studies*, Cunningham P, Cunningham M (eds.), IOS Press, 2008.
3. Silaghi GC, Arenas A, Silva LM. A Utility-Based Reputation Model for Service-Oriented Computing. *Toward Next Generation Grids*, Priol T, Vanneschi M (eds.), CoreGRID Series, Springer, 2007; 63–72.
4. Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications* 2001; **15**(3):200–222.
5. Jøsang A, Ismail R, Boyd C. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems* March 2007; **43**(2):618–644.
6. Abdul-Rahman A, Hailes S. Supporting Trust in Virtual Communities. *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, IEEE Computer Society, 2000; 6007.
7. Silaghi GC, Arenas A, Silva LM. Reputation-based trust management systems and their applicability to grids. *Technical Report TR-0064*, Institutes on Knowledge and Data Management and System Architecture, CoreGRID - Network of Excellence February 2007.
8. Zacharia G, Maes P. Trust management through reputation mechanisms. *Applied Artificial Intelligence* 2000; **14**(9):881–907.
9. Gupta M, Judge P, Ammar M. A reputation system for peer-to-peer networks. *NOSSDAV '03: Proc. of the 13th international workshop on Network and operating systems support for digital audio and video*, ACM Press, 2003; 144–152.
10. Aberer K, Despotovic Z. Managing Trust in a Peer-2-Peer Information System. *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, ACM Press, 2001; 310–317.
11. Singh M, Yu B, Venkatraman M. Community-based service location. *Commun. ACM* 2001; **44**(4):49–54.
12. Despotovic Z, Aberer K. Probabilistic Prediction of Peers' Performance in P2P Networks. *Engineering Applications of Artificial Intelligence* 10 2005; **18**(3):771–780.
13. Sherwood R, Seungjoon L, Bhattacharjee B. Cooperative peer groups in nice. *Computer Networks* 2006; **50**(4):523–544.
14. Sabater J, Sierra C. REGRET: A Reputation Model for Gregarious Societies. *Fourth Workshop on Deception, Fraud and Trust in Agent Societies*, ACM Press, 2001; 194–195.
15. Ramchurn S, Jennings N, Sierra C, Godo L. Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence* 2004; **18**(9-10):833–852.
16. Huynh T, Jennings NR, Shadbolt N. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 2006; **13**(2):119–154.
17. Jøsang A, Knapskog S. A metric for trusted systems. *Proceedings of the 21st National Information Systems Security Conference, (NIST-NCSC 1998)*, 1998.
18. Marsh S. Formalizing trust as a computational concept. PhD Thesis, University of Stirling 1994.
19. Jøsang A, Ismail R. The Beta Reputation System. *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.
20. Mui L, Mohtashemi M, Halberstadt A. A Computational Model of Trust and Reputation for E-businesses. *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7*, IEEE Computer Society, 2002; 188.
21. Buchegger S, Le Boudec JY. A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. *P2PEcon 2004*, 2004.
22. Teacy WTL, Patel J, Jennings NR, Luck M. Coping with Inaccurate Reputation Sources: Experimental Analysis of a Probabilistic Trust Model. *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, ACM Press, 2005; 997–1004.
23. Yu B, Singh M. An evidential model of distributed reputation management. *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM Press, 2002; 294–301.
24. Kamvar S, Schlosser M, Garcia-Molina H. The eigentrust algorithm for reputation management in p2p networks. *WWW '03: Proceedings of the 12th international conference on World Wide Web*, ACM Press, 2003; 640–651.
25. Singh A, Liu L. Trustme: Anonymous management of trust relationships in decentralized p2p systems. *P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, IEEE Computer Society, 2003; 142–149.
26. Xiong L, Liu L. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions*

on Knowledge and Data Engineering 2004; **16**(7):843–857.
27. Jurca R, Faltings B. An incentive compatible reputation mechanism. *2003 IEEE International Conference on E-Commerce Technology*, IEEE Computer Society, 2003; 285.
28. Jurca R, Faltings B. Reputation-based pricing of p2p services. *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, ACM Press, 2005; 144–149.
29. Jurca R, Faltings B. Reputation-based service level agreements for web services. *ICSOC*, 2005; 396–409.
30. Anderson DP. Boinc: A system for public-resource computing and storage. *5th International Workshop on Grid Computing (GRID 2004), Pittsburgh, PA, USA, Proceedings*, Buyya R (ed.), IEEE Computer Society, 2004; 4–10.
31. Sarmenta LFG. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Gener. Comput. Syst.* 2002; **18**(4):561–572.
32. Zhao S, Lo V, GauthierDickey C. Result verification and trust-based scheduling in peer-to-peer grids. *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, IEEE Computer Society, 2005; 31–38.
33. Silaghi GC, Silva LM, Domingues P, Arenas AE. Tackling the Collusion Threat in P2P-Enhanced Internet Desktop Grids. *Proceedings of CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments*, CoreGRID Series, Springer, 2008.
34. von Laszewski G, Alunkal B, Veljkovic I. Towards reputable grids. *Scalable Computing: Practice and Experience* 2005; **6**(3):95–106.
35. Kerschbaum F, Haller J, Karabulut Y, Robinson P. PathTrust: A Trust-based Reputation Service for Virtual Organization Formation. *iTrust2006: Proceedings of the 4th International Conference on Trust Management*, *Lecture Notes in Computer Science*, vol. 3986, Springer, 2006; 193–205.
36. Camarihna-Matos LM, Afsarmanesh H. Elements of a base ve infrastructure. *Journal of Computers in Industry* 2003; **51**(2):139–163.
37. Woodcock J, Davies J. *Using Z: Specification, Refinement, and Proof*. Prentice-Hall, Inc., 1996.
38. Legrand A, Marchal L, Casanova H. Scheduling Distributed Applications: the SimGrid Simulation Framework. *CCGRID '03: Proc. of the 3st International Symposium on Cluster Computing and the Grid*, IEEE Computer Society, 2003; 138.

DOI: 10.1002/sec