ADAPTED QUALITY RESOURCE SELECTION USING THE GRID REPUTATION-POLICY TRUST MANAGEMENT SERVICE

Yonatan Zetuny
PhD Researcher
Centre for Parallel Computing, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK

Dr. Gabor Terstyanszky
Reader
Centre for Parallel Computing, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK

Prof. Stephen Winter
Professor
Centre for Parallel Computing, University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK

Prof. Peter Kacsuk
Professor
MTA SZTAKI, 1111 Kende utca 13, Budapest, Hungary

ABSTRACT

The concept of trust and reputation had a long profound impact on the way systems were measured in terms of their credibility and performance. This has consecutively lead into numerous reputation-based trust management systems being deployed for various computing environments as a decision support tool for assessing the trustworthiness of participating parties. In the context of Grid computing, reputation-based trust management systems play an important role for supporting coordinated resource sharing as they can reduce job execution failure by selecting relatively competent resources based on aggregated historical recommendations and given job requirements. In this paper, we present the Grid reputation-policy trust management service (GREPTrust) for managing resource selection in computational grids. This infrastructure level service encompasses a novel *reputation-policy* trust model, which enables service consumers (e.g. monitoring toolkits and resource brokers), to carry out an active participation in the trust and reputation evaluation processes. This is achieved by enabling service consumers to augment standard reputation queries with a set of reputation-policy statements rectified as trust decision strategies. Each strategy forms complete trust metrics blueprint for the reputation algorithm and therefore allows fine-grained resource selection adapted to specific job requirements.

KEYWORDS

Grid Computing, Security, Trust, Reputation, Policy, Qos.

1. INTRODUCTION

The Grid computing research was initiated as a way of supporting scientific collaborations (I. Foster, 2001). Grid systems were mainly used in e-science projects where resources from trusted organizations were pooled together in order to collaborate and form the Grid. Trust between different entities in the Grid was initially addressed through security mechanisms (V. Welch, 2003). These mechanisms enabled single sign-on (SSO) for an entity in the system, considering that the entities belonged to the same trusted domain. Authentication mechanisms were provided through certificates (e.g. x.509 proxy certificate) which entitled the nodes belonging to the trusted organizations to join the Grid (I Foster, 2002). Consequently, trust creation and

management involved an inflexible human-coordinated process. This form of *certification trust* was satisfactory for static Grids where the set of users and resources remains constant during computations.

However, as the Grid shifted to ubiquitous and pervasive computing models, there became an increasing demand to be able to dynamically establish *behavioural trust* relationships between all entities *after* they joined the Grid (G. Cosmin Silaghi, 2007). This form of trust determines the degree by which a trusting agent incrementally trusts a trusted agent based on the accumulation of information gathered by the trusting agent regarding the trusted agent behaviour over time (M. Brinkløv, 2007). The accumulated information is typically made up of two sources: **direct trust** and **indirect trust**. Direct trust is based on the personal experience a trusting agent has achieved with a trusted agent whereas indirect trust is evaluated from ratings provided by other trusting agents regarding their personal experiences with the trusted agent. The literature classifies reputation-based trust management systems in two main categories: systems that solely use direct trust measures and systems which use both direct and indirect trust measures (G. Cosmin Silaghi, 2007).

Trust and security research initiatives have produced three predominant methodologies for modelling reputation: *discrete models*, methods which employ *Bayesian statistics* and methods which are based on *fuzzy logic*. Each methodology utilizes a different approach for combining direct and indirect trust for synthesizing the trustworthiness value produced by recommendation agents' opinions. Discrete models use linear combinations of direct and indirect trust functions to evaluate the trust $T(x, y, t)$ of trusting agent x in trusted agent y at time t. The Bayesian methodology uses a probabilistic approach to the determination of the reputation. It uses the known Bayes formula to determine the conditional probability $p(H|E)$ based the occurrence of event H given symptom E. The fuzzy logic methodology perceives trust as a linguistic concept which is poorly described by numerical values. It uses fuzzy inference for formulating the mapping of a given input behavioural parameter to an output using membership functions, logical operations and conditional rules. These reputation modelling methodologies form the basis logic behind different reputation-based trust management systems available in different distributed computing environments, such as electronic markets, P2P systems and Grid computing. At present, there are only few approaches for reputation models applied to Grid computing, despite their merits for improving resource allocation and scheduling. Existing Grid trust management systems are generally based on earlier developments and requirements identified in P2P systems and they are focused on resource management and security enhancements. For example, GridEigenTrust (B. Alunkal, 2003) and PathTrust (F. Kerschbaum, 2006) were one of the first attempts to incorporate reputation-based trust management systems into Grid computing environments. Both solutions apply probabilistic approach to the determination of resource reputation. Current Grid reputation-based trust models propose a central reputation service providing deterministic, predefined metrics for selecting a trusted resource from a list of possible alternatives. These approaches are limited as that they do not allow external involvement in the trust and reputation evaluation processes. Grid clients are not able to calculate the trust value of a Grid resource by specifying their own reputation evaluation criteria and as a result, they have to rely on a central reputation algorithm to compute trust values.

These limitations stimulated the motivation behind the *reputation-policy* trust model for Grid resource selection (Y. Zetuny, 2008). This model allows carrying out an active participation in the trust and reputation evaluation processes. This is achieved by enabling Grid client applications to augment their existing reputation queries with a set of reputation-policy statements rectified as a trust decision strategy. The strategy is comprised of an evaluation model represented by a set of opinions (or quality aspects) as well as a set of decision rules describing potential trust value calculations and correspondent actions. Together they form complete trust metrics blueprint for the reputation algorithm. The philosophy behind the reputation-policy trust model is that reputation is perceived as *subjective* matter and context dependent. This implies that Grid clients should be supplied with means to evaluate Grid resources by articulating subjective trust based decisions reflecting their intrinsic view on trust and the type of job they wish to submit. The Grid Reputation-Policy Trust management service (GREPTrust) provides a reference implementation for reputation-policy trust model (Y. Zetuny, 2008). The aim of this paper is to demonstrate GREPTrust advantages over existing reputation models when resource selection based on evaluation criteria is crucial for successful computation. This is achieved by constructing a testbed experiment which compares GREPTrust with GridPP UK SAM test results (GridPP, 2009) when selecting computing nodes (CPU cycles) for a financial simulation.

This paper is structured as follows. Section 2 briefly describes the core concepts of the reputation-policy trust model and its main artefacts. Section 3 describes the GREPTrust architecture and query management facilities. Section 4 describes testbed selection & implementation, Section 5 scenario analysis and experiment results. Finally, section 6 discusses conclusions, including limitations and potential applications.

2. REPUTATION-POLICY TRUST MODEL

The foundation contribution presented in this paper is prompted by the reputation-policy trust model. This reputation model is based on the theoretical work made by (Gambetta, 1988) and envisions trust as *subjective* belief a trusting agent has in the capability of a trusted agent to deliver a quality service for a given job context and time slot. In practical terms, the reputation-policy trust model is a *reputation model* which allows trusting agents to query aggregated historical *recommendations* made by recommendation agents regarding the *trustworthiness* of trusted agents. In similar manner to (J. Jia, 2006), trustworthiness is perceived in this model as *multi-faceted* consisting of an aggregation of quality aspects such as availability, reliability, data accuracy and etc. The aggregated historical recommendations contain testimonies for different quality factors which blend both *direct* and *indirect* previously achieved experiences with the different trusted agents. The most predominant addition of the reputation-policy trust model lies in the fact that trusting agents are able to manipulate the aggregated historical recommendation feedbacks and adapt the retrieved data for their own *subjective* service/product needs. This is achieved by introducing three artifacts: Trust Decision Strategy (TDS), Opinion Matrices (OM) and the Correlation Process (CP), which are illustrated in Figure 1.

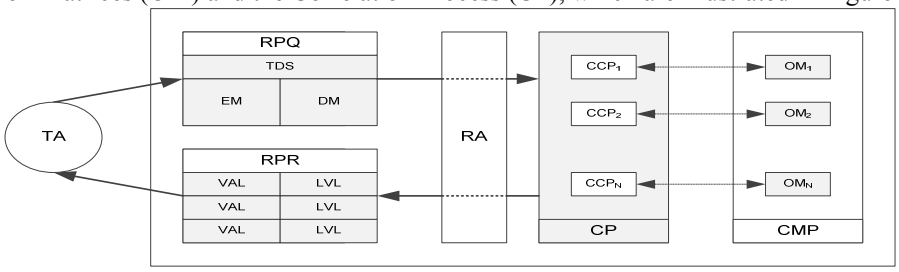

Figure 1. Reputation-Policy Trust Model

The TDS is comprised of an *Evaluation Model* (EM) and a *Decision Mode*l (DM). The EM allows trusting agents to stipulate reputation-policy statements that control which quality aspects to process, the weight factors to consider between these aspects, the sources to blend (direct, indirect or both) for each quality aspect and the weight factors between these sources. The DM is based on a *Fuzzy Inference System* (FIS) (IEC 1131, 1997) and consists of reputation-policy statements which stipulate mapping rules for translating trust values produced by the EM into trust levels, which are known as trust decisions. The mapping rules consist of *fuzzification*, *defuzzification* and *decision rules*. The fuzzification aspect reads the trust values of the input variables and converts them into degrees in predefined membership functions. The defuzzification is the process of producing a quantifiable trust level using the decision rules and applying a defuzzification method such as Centre of Gravity (CoG) or Middle of Maximum (MoM). The TDS is appended to a Reputation-Policy Query (RPQ) and submitted to the *Reputation Algorithm* (RA). Since the trust metrics are defined in the TDS, the RA merely functions a dispatcher to the *Correlation Process* (CP). The CP is modeled as a multithreaded algorithm. For each quality aspect defined in the EM the CP spawns a *Child Correlation Process* (CCP). The role of the CCP is to correlate between the requirements associated with a single quality aspect (i.e. weight factor, sources and etc) and the data managed by a dedicated Opinion Matrix (OM) inside a *Customized Metrics pool* (CMP). The CMP contains a collection of Opinion Matrices (OM) – one for each type of quality aspect (i.e. availability, reliability and etc). Each OM contains an aggregation of recommendation feedback values ranging between [0, 1] made by each recommendation agent regarding the trustworthiness of each trusted agent. Each CCP operates on its correspondent OM and returns the quality-aspect trust value back to the CP. Once all CCP completed their tasks, the CP iterates through each value, applies its associated weight factor and calculates the final trust value using a weighted mean formula. The correlation process routes each trust value together with the DM segment into an FIS processor, which in turn calculates the correspondent trust level. Finally, the correlation process collects each trust value/level pair, generates a *Reputation-Policy Report* (RPR) and submits the report to the trusting agent. The RPR contains an array where each element represents an evaluated trusted agent, and its trust value/level pair. The trusting agent inspects the report and initiates an interaction with the relevant trusted agent(s).

3. GRID REPUTATION-POLICY TRUST MANAGMEMENT SERVICE ARCHITECTURE

The consecutive contribution presented by this paper is to provide a reference implementation for the reputation-policy trust model and apply it in a Grid computing environment. This involves analyzing the Grid requirements towards trust and reputation and deriving a service architecture which incorporates the various artifacts of the model into a single self-contained Grid reputation service known as the *Grid Reputation-Policy Trust Management Service* (GREPTrust). While intended to support VOs during the operation phase, GREPTrust provides three major functions for the participating entities: capturing reputation-policy requirements, computing resources trustworthiness and evaluating resources after transactions. Figure 2 illustrates the GREPTrust architecture, which is comprised of three underlying subsystems: **Client system** (Grid Client, TDS Data Store), **Service system** (Reputation-Policy Service Facade, Querying Manager, Feedback Manager and Admin Manager) and **Data system** (Reputation-Policy Data Store). The remainder of this paper is concentrated on the Grid resource querying mechanism provided by the Querying Manager.

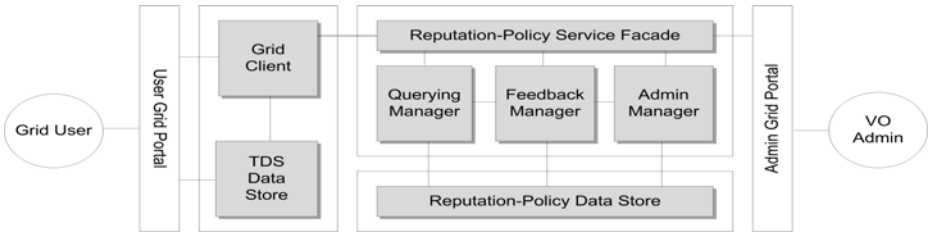

Figure 2. GREPTrust Management Service Architecture

The *Querying Manager* (QM) component is responsible of Grid resource querying facilities as well enabling aggregated reputation queries used for discovering the reputation of organizations based on the reputation of their underlying resources. It can even be used for discovering the reputation of an entire VO by calculating the reputation of its underlying organizations. The QM enables Grid clients to evaluate resources based on the reputation evaluation criteria they supply to it. Grid clients are required to submit a *Reputation-Policy Query* (RPQ) to the QM, which contains the *Trust Decision Strategy* (TDS) as well as context information data such as client id, resources to evaluate, cut-off date time and a trust decay function. The QM processes the RPQ and generates a *Reputation-Policy Report* (RPR) which is returned to the Grid client. The Grid client analyzes the report in order to make a final decision on which resource(s) it should submit its jobs to. The Grid resource querying mechanism is divided into three steps (*Process TDS Evaluation Model*, *Process TDS Decision Model* and *Generate Reputation-Policy Report*). Figure 3 illustrates the internal architecture of the QM and table 1 summarizes the three steps required for processing an RPQ.

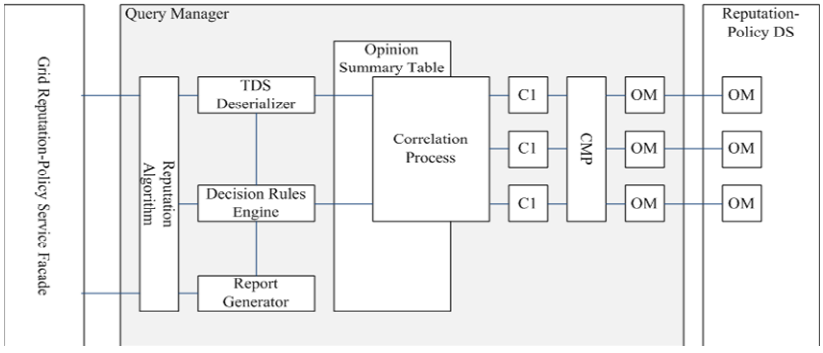

Figure 3. Querying Manager Architecture

Table 1. RPQ Processing Algorithm Summary

Algorithm 1: RPQ Processing Algorithm(RQC)

RPQ:	reputation query context	RESK:	resource key
TDSXML:	TDS XML string	RESV:	resource value
TDSPRS:	TDS parser	RT:	rule table
TDS:	trust decision strategy	RK:	rule key
LOPN:	list of opinions	RV:	rule value
CP:	correlation process	RD:	rule degree
RVT:	resource value table	RF:	report factory
TDRENG:	trust decision rule engine	R:	report
TDM:	trust decision model	RPRS:	report parser
RES:	resource	RXML:	report XML string

```
/* Process TDS Evaluation Model */
TDSXML ← RPQ.getTrustDecisionStrategy()
TDSPRS ← new()
TDS ← TDSPRS.fromXML(TDSXML)
LOPN ← TDS.getTrustEvaluationModel().getOpinions()
CP ← new(LOPN, RQC)
RVT ← CP.process()

/* Process TDS Decision Model */
TDM ← TDS.getTrustDecisionModel()
if TDM != null then
    TDRENG ← new(TDM)
end

R ← RF.getReport()
foreach RESK in RVT.keys do
    RESV ← RVT.get(RK)

    // default behaviour is to assign resource level as resource value.
    RES ← R.addResource(RESK, RESV, RESV)
    if TDM != null then
        // process trust decision rules engine.
        RT ← TDRENG.process(RESV)

        foreach R in RT.keys do
            RD ← RT.get(RK)
            if (RK == "trust_level") then
                RV ← RT.get(RK)
                RES.setLevel(RV)
                continue
            end
            RES.addRule(RK, RD)
        end
    end
end

/* Generate Reputation-Policy Report */
RPRS ← new()
RXML ← RPRS.toXML(R)

return RXML
```

4. GREPTRUST TESTBED IMPLEMENTATION

The selection of an adequate testbed is of paramount importance for the validation of GREPTrust and the success of this research. The main requirement from such testbed is to allow efficient deployment of the model as a Grid service and simulate interactions between Grid clients, GREPTrust and Grid resources. The intention of the testbed is to provide an infrastructure for an automated testing framework for comparing Grid reputation models as well as demonstrating how the Grid reputation-policy trust model improves resource selection in mission critical scenarios. The selected infrastructure for implementing the reputation-policy trust model is based on the GridSim simulation toolkit (R. Buyya, 2002). Essentially, the GREPTrust testbed environment was designed for comparing the GREPTrust reputation model with an existing reputation model and proving the advantages of stipulating reputation requirements when qualifying resources for selection. The comparison to an existing model is not quite feasible to conduct as the GREPTrust model is based on heuristics as opposed to the deterministic models offered by other reputation solutions. The reason for that is that GREPTrust trust metrics are stipulated externally in the TDS file whereas existing reputation models rely on a central reputation algorithm bounded to esoteric trust metrics which do not support external manipulation. In order to overcome this challenge, it has been decided to deploy two instances of GREPTrust. The first instance represents a heuristic approach where it is able to accept a different TDS file in each execution depending on the historical ratings data, VO characteristics, the job requirements and risk tolerance - all acquired prior to any execution. The second GREPTrust instance represents a deterministic approach. It is modelled after the reputation algorithm used in the GridPP project (GridPP, 2009). This instance uses a constant TDS which contains a collection of a single opinion (availability), equal weight rules and no decision model thus implicitly implying that trust values are equal to trust levels. In practical terms, this instance computes resources trust values using an arithmetic mean of their historical percentage of availability time. Figure 4 illustrates the GREPTrust testbed architecture. It is comprised of four logical components: *GREPTrustAnalysis*, *GREPTrustAPI*, *GREPTrustTestBed* and *GREPTrustDB*. The central component is GREPTrustTestbed. Implemented on top of the GridSim environment, it contains two GREPTrust resources (heuristic and deterministic). Both resources access the GREPTrustAPI component, which contains the core querying logic and accessibility to the historical ratings feedback data residing in the GREPTrustDB component. GREPTrustTestBed also contains the simulation logic which initializes the environment, instantiates the two GREPTrust resources and a generic Grid client which constructs two reputation-policy queries, submits each query to a GREPTrust resource and stores the resource evaluation reports in the file system. The GREPTrustAnalysis package wraps the entire testbed environment. It accommodates an automated testing framework for executing different simulation scenarios. This framework uses Python configuration scripts to prepare each scenario simulation (e.g. preload the GREPTrustDB with extracted usage data), control the GREPTrust testbed using the JPype bridge library and plot simulation results to Matlab. In order to ensure the accuracy of the comparison between the two models, an identical historical dataset was used. While it is possible to generate ratings feedback data, the preferred way is to use real sample data. For the simulation purposes, data was retrieved from the nodes used for the GridPP project (UK SAM Test Results). The data contains both availability as well as reliability information for each of the participating Grid resources. Since the GREPTRust database stores ratings feedback data which is derived from performance data (but not actually performance data), simulation scripts will generate reputation data out of the performance ratio of each Grid resource and populate the relevant tables in GREPTrust database.

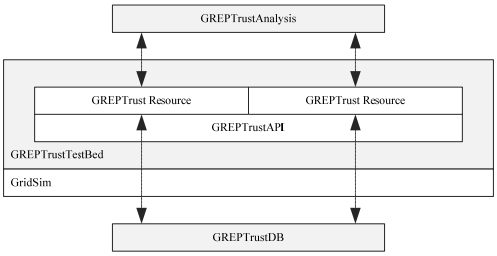

Figure 4. GREPTrust Testbed Architecture Depicting the Four Logical Components of the GREPTrust Testbed

5. CASE STUDY EXPERIMENTS & EVALUATION

The purpose the testbed experiment is to provide an evaluation of the reputation-policy trust model through a real-world financial case study. The topic for the case study is taken from an increasingly popular field in computational finance known as *algorithmic trading*. In electronic financial markets, the term algorithmic trading refers to the utilization of computer programs for entering trading orders with the computer algorithm deciding on certain aspects of the order such as the timing, price and the final quantity of the order (I. Domowitz, 2005). Sell-side vendors (e.g. brokers) offer execution-type algorithms intended for slicing orders (i.e. large block of shares) submitted by institutional traders and efficiently ensuring that each sliced order is getting the best possible price with minimum market impact. Trading algorithms encapsulate complex quantitative models which normally rely on historical analytics for determining the algorithm behaviour. In this particular scenario, we focus on historical volume distribution generation for a VWAP algorithm. Primarily used by institutional traders, the VWAP algorithm is a schedule-driven strategy which processes orders over specified time horizon, spreading each trade in proportion to the historical volume distribution resulting in minimized market impact (S. M. Kakade, 2004). The algorithm performance is measured against the VWAP benchmark which is defined as the ratio of the value traded (price times shares traded) to the total shares traded for a given day. Evaluating the performance of a VWAP algorithm is straight forward – if the price of a buy order is lower than the VWAP, the trade is considered good; alternatively, if the price is higher - it is considered poor. High sensitivity to accurate generation of volume patterns and impact on strategy performance make historical volume distribution an ideal candidate for reputation evaluation experiments. Leveraging on the GREPTrust testbed, we designed a simulation experiment for volume distribution generation and compared both instances of GREPTrust (heuristic and deterministic) to determine which model selects higher quality computing resources for calculating volume distributions and eventually resulting in higher performing strategy instances. The volume distribution is calculated by sub tasking a predefined stock universe between 22 target computing resources. Each resource iterates through each stock symbol and partitions the stock trading day into 42 bins where each bin contains 21 days average volume percentage for execution during 15 minutes time frame. Being sensitive to data quality, the heuristic TDS was prepared with 50/50 ratio balance on availability/data accuracy opinions. The deterministic model remained on a single opinion (availability). Figure 5 illustrates a plotting example in Matlab where the two GREPTrust instances are being compared before executing the testbed experiment. The horizontal axis lists Grid resources 1 to 22 and the vertical axis denotes the trust level. The darker and the lighter series points denote GridPP and GREPTrust respectively. This example illustrates how the decision model in the TDS could be used when the Grid client has previously acquired negative impression on the VO characteristics (e.g. unreliable Grid) and set the trust level 0.7 as minimum threshold. The decision model is being used in the GREPTrust example to harden the criteria for matching competent resources and as the graph shows, the resources evaluated by GREPTrust have comparably lower trust levels than the ones evaluated by GridPP. Out of 22 resources, only 11 resources were reported competent enough for selection by GREPTrust. In contrast, 21 resources were reported competent by GridPP. This demonstrates how one can greatly reduce the risk of job execution failure using external reputation evaluation criteria.

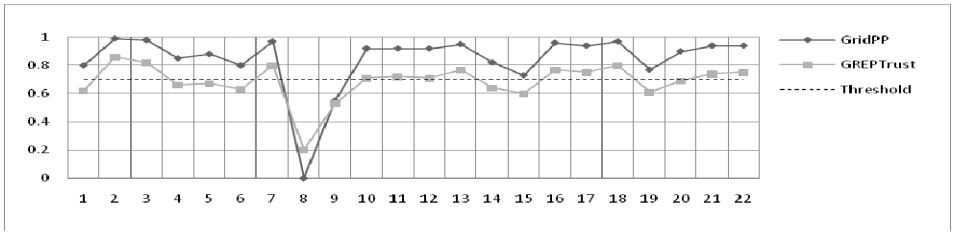

Figure 5. Grid Reputation Models Comparison used for Generating Historical Volume Distributions

6. CONCLUSIONS

This paper presented a novel architecture for managing trust in Grid computing. While existing esoteric solutions offer a single, community-based deterministic reputation algorithm for computing trust values, the

Grid reputation-policy trust management service (GREPTrust) promotes external involvement in the trust and reputation evaluation processes using well defined reputation-policy statements. This allows fine-grained resource selection based on a trust decision strategy (TDS) adapted to specific job/service requirements. Being an infrastructure-level service, GREPTrust can be integrated with resource brokers or monitoring toolkits for enhancing their functionalities. Based on a practical financial case study, we conducted quantitative experiments with GREPTrust and proved its probability to select higher quality resources and lower the risks for faulty computations. Despite the overhead of generating trust metrics at the client side and the complete reliance on historical data (currently no integration with real-time performance data nor trust prediction model), we firmly believe that GREPTrust can significantly enhance Grid user experience and satisfaction and in the longer term, this may further increase Grid computing adaptation in the industry and especially in the financial markets where data accuracy/quality is a highly crucial matter.

REFERENCES

B. Alunkal I. Veljkovic, G. von Laszewski, K. Amin Reputation-Based Grid Resource Selection [Conference] // Workshop on Adaptive Grid Middleware. - 2003.

EGEE Project http://glite.web.cern.ch/glite/ [Online]. - 2009.

F. Kerschbaum J. Haller, Y. Karabulut, P. Robinson Pathtrust: A trust-based reputation service for virtual organization formation [Conference] // iTrust2006, 4th International Conference on Trust Management. - [s.l.] : Springer, 2006. - Vol. 3986.

G. Cosmin Silaghi A. E. Arenas, L. M. Silva Reputation-based trust management systems and their applicability to grids [Report]. - [s.l.] : CoreGRID TR-0064, 2007.

Gambetta D. Can We Trust Trust? [Book Section] // Trust: Making and Breaking Cooperative Relations. - Oxford : University of Oxford, 1988.

GridPP http://pprc.qmul.ac.uk/~lloyd/gridpp/samtest.html [Online] // GridPP. - UK Computing for Particle Physics, 2009.

I Foster C Kesselman, J Nick, S Tuecke The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration [Conference] // Global Grid Forum. - [s.l.] : Open Grid Service Infrastructure WG, 2002.

I. Domowitz H. Yegerman The Cost of Algorithmic Trading: A First Look at Comparative Performance [Journal]. - [s.l.] : ITG Research, 2005.

I. Foster C. Kesselman, S. Tuecke The Anatomy of the Grid: Enabling Scalable Virtual Organizations [Journal]. - [s.l.] : International Journal of High-Performance Computing Applications, 2001. - 3 : Vol. 15.

J. Jia X. Qu Towards Trustworthy Resource Selection in Grid: A Fuzzy Partial Ordering based Approach [Journal]. - Seoul : IJCSNS International Journal of Computer Science and Network Security, 2006. - 9A : Vol. 6.

M. Brinkløv R. Sharp Incremental Trust in Grid Computing [Conference] // Seventh IEEE International Symposium on Cluster Computing and the Grid. - [s.l.] : CCGrid 07, 2007.

R. Buyya M. Murshed GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing [Report]. - 2002.

S. M. Kakade M. Kearns, Y. Mansour L. E. Ortiz Competitive Algorithms for VWAP and Limit Order Trading [Journal]. - New York : EC'04, 2004.

V. Welch F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski et al. Security for Grid Services. [Conference] // In Proceedings of 12th IEEE International Symposium on High Performace Distributed Computing.. - [s.l.] : IEEE Computer Society Press, 2003.

Y. Zetuny G. Terstyanszky, S. Winter, P. Kacsuk Articulating Subjective Trust-Based Decision Strategies Utilizing the Reputation-Policy Trust Management Service [Conference] // UK e-science 2008 All Hands Meeting. - Edinburgh : [s.n.], 2008.

Y. Zetuny G. Terstyanszky, S. Winter, P. Kacsuk Reputation-Policy Trust Model for Grid Resource Selection [Conference] // DAPSYS2008. - Debrecen : 7th International Conference on distributed and parallel systems, 2008.