A synergistic reputation-policy based trust model for Grid resource selection.

Yonatan Zetuny

School of Electronics and Computer Science

A Synergistic Reputation-Policy Based Trust Model for Grid Resource Selection

A dissertation submitted to the University of Westminster for the degree of Doctor of Philosophy

Yonatan Zetuny

Supervised by Dr. G. Z. Terstyanszky, Prof. S. C. Winter & Prof. K. Madani

February 2011

Centre for Parallel Computing,
School of Electronics and Computer Science,
University of Westminster,
London, United Kingdom

To my loving parents, Eliyahu and Malca Zetuny

Acknowledgements

This thesis is the culmination of a long journey throughout which I have received support from various professionals whom I wish to acknowledge here. First and foremost, I am deeply indebted to my research supervisor, Dr. Gabor Terstyanszky, for his advice and for fuelling my motivation and enthusiasm during the last six years.

I would like to express my gratitude towards Prof. Stephen Winter, my director of studies, and for Prof. Peter Kacsuk for their intriguing questions, encouragement, support and advice that I have received from them during my candidature.

I would like to thank Prof. Kambiz Madani and the University of Westminster research committee for admitting me to the PhD programme, granting me the research scholarship and assisting me during the initial steps of my doctoral studies.

I would like to thank my former work colleagues, Mr. Eyal Zaidman, Mr. Ofer Guez and Mr. Joe Eldridge for their endless support, suggestions and ideas which contributed to the research quality.

I would like to express my gratitude towards my dear friend, Mr. Taher Majidi for helping me gain professional recognition as well as supporting me emotionally when I mostly needed to.

Last but not least, I would like to express my deepest gratitude towards Mr. Effie Nadiv for introducing me to the exciting world of computer science. I owe my decision to pursue a professional career as a computer scientist to his ability to spark my tremendous desire to evolve in this remarkable field.

"If you want something, go get It. Period."
(The Pursuit of Happyness, 2006)

Yonatan Zetuny, February 2011
London, United Kingdom

Abstract

In the context of Grid computing, reputation-based trust management systems are playing an increasingly important role for supporting coordinated resource sharing and ensuring provision of quality of service. However, the existing Grid reputation-based trust management systems are considered limited as they are bounded to esoteric reputation-based trust models encompassing predefined metrics for calculating and selecting trusted computing resources and as a result, they prevent external involvement in the trust and reputation evaluation processes.

This thesis suggests an alternative approach for reputation modelling founded on its core argument proclaiming that reputation is a subjective matter as well as context dependent. Consequently, it offers a synergistic *reputation-policy* based trust model for Grid resource selection. This exoteric trust model introduces a novel paradigm for evaluating Grid resources, in which Grid client applications (e.g. monitoring toolkits and resource brokers) are endeavoured to carry out an active participation in the trust and reputation evaluation processes. This is achieved by augmenting the standard reputation queries with a set of reputation-policy assertions constituting as complete trust metrics supplied into the reputation algorithm. Consecutively, the Grid Reputation-Policy Trust management system (GREPTrust) provides a concrete implementation for the trust model and it's underlying artifacts whilst the GREPTrust testbed provides an adequate infrastructure for comparing the reputation-policy trust model with a production available esoteric model (GridPP).

Based on a computational finance case study, an internal workflow simulation utilises the GREPTrust testbed in order to empirically assess the criteria by which the synergistic reputation-policy based trust model outperforms esoteric trust models regarding resource selection and consequently provides substantive evidence that the reputation-policy paradigm is a welcome addition to the Grid computing community.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

v	Trust value	32
l	Trust level	32
S_e	Direct experience trust source	32
S_r	Indirect (reputation) trust source	32
$\mathbb{P}(S)$	Power set of S, where $S = \{S_e, S_r\}$	32
x	Trusting agent x	35
y	Trusting agent y	35
z	Domain of entities excluding trusting agent x	36
t	Time fragment t	35
t_{xy}	Time of last transaction between trusting agent x and trusted agent y	35
c	Context c	35
$\sum_{k=1}^{n} P_r(y)$	Number of positive feedbacks for trusted agent y	35
$\sum_{k=1}^{n} N_r(y)$	Number of negative feedbacks for trusted agent y	35
$\Gamma(y)$	General trust function	35
$\Gamma(x,y,t,c)$	Discrete trust function	35
$\Theta(x,y,t,c)$	Direct trust function	35
$\Omega(y,t,c)$	Indirect (reputation) trust function	35
α	Direct trust coefficient	35
β	Indirect (reputation) trust coefficient	35
$\Upsilon(t-t_{xy},c)$	Trust decay function	35
$R(z,y)$	Recommender trust factor	36
H	Event H	36
E	Symptom E	36
$P(H \mid E)$	Conditional probability P based on the occurrence of H given E	36
$\mu(x_1)$	fuzzy degree of membership for input variable x_1	37
$\mu_{Ok}(y)$	Membership function for fuzzy output sets O_k,k = 1,...,r	38
$y*$	Output value based on fuzzy reasoning	38
R	A list of evaluated resources	53
$\#R$	Cardinality of R	53
O	Opinion	76

Chapter 1

Introduction

"A fact is a simple statement that everyone believes. It is innocent, unless found guilty. A hypothesis is a novel suggestion that no one wants to believe. It is guilty, until found effective."

Edward Teller

1.1 Introduction

The purpose of this chapter is to introduce the framework for the research. It is therefore comprised of four sections: research overview, trust management concepts and challenges, research deliverables and thesis structure. The research overview section describes the problem statement and the objective of this research. The trust management concepts and challenges section describes the key terms used throughout this thesis (e.g. trust, reputation and policy) and consecutively describes the research challenges. The research deliverables section lists the research contributions. The thesis structure section describes the contents of this thesis.

1.2 Research Overview

The concept of trust and reputation had a long profound impact on the way systems were measured in terms of their credibility and performance [21, 22]. This has consequently led into numerous reputation-based trust management systems being deployed for various computing environments (such as P2P and electronic markets [48]) as a decision support tool for assessing the *trustworthiness* of participating parties [70, 27, 29]. In the context of Grid computing [7, 8], reputation-based trust management systems play an important role for supporting coordinated resource sharing, as they can manage job execution risk by preemptively selecting computing resources based on aggregated historical recommendations [50, 51, 52].

Despite the increasing popularity of Grid computing within several industrial sectors [12, 140, 104, 141] and the rising number of commercial Grid solutions being offered by leading IT vendors [136, 137, 138, 139], reputation still remains a major concern, as it is essential for establishing trust dynamically between untrusted parties (such as newly formed business partnerships). In fact, reputation is considered as one of the tools that the research community is required to supply in order expand the Grid beyond the organisational boundaries and gain wide acceptance by various industries. In addition, the usage of reputation models can reduce the gap that currently exists between classical grids and desktop grids, making desktop grids trustworthy and allowing them to be used as the classical grids are [52].

Considering the numerous types of jobs that can be submitted to the Grid by end users in numerous industrial sectors (e.g. engineering, medicine, biology, finance and etc.) raises concerns regarding the applicability of a single reputation-based trust management system to cater for any type of job requirement. In turn, this leads to the main hypothesis suggesting that a reputation model which considers external evaluation factors such as, job requirements, global context and risk attitude may optimise resource selection and hence - improve the overall quality of computations.

1.2.1 Problem Statement

The Grid computing research was initiated as a way of supporting scientific collaborations [9]. Grid systems were mainly used in e-science projects [121, 123, 124] where resources from trusted institutions were pooled together in order to collaborate and form the Grid [125, 126]. However, as Grid systems increasingly gained popularity for supporting business operations [118, 119, 120], a growing demand has been raised on how to share computing resources between potentially unknown parties [52]. As a result, dynamic trust establishment has become a crucial factor for qualifying resources for selection, thus ensuring successful business collaborations.

Despite the acknowledgement over the importance of trust management in Grid computing [50], reputation based trust models are still barely considered for classical Grid systems [52]. Since the long term future of the Grid is to provide dynamic aggregation of resources, provided as services between businesses, new architectures and detailed mechanisms for bringing together arbitrary resources are required [9]. Reputation-based trust management systems are projected to fulfil this requirement by adding an important value for scheduling and executing Grid workflows consisting of trustworthy resources which had been previously evaluated and selected based on the aggregation of recommendations regarding their different quality factors [52].

This thesis argues that the deployment of a reputation-policy based trust management system is a welcome addition to the Grid computing community as it would allow an elevated degree of fine-grained resource selection in order to exoterically

match specific job requirements. As a result, this makes the reputation-policy trust research topic an important milestone toward the evolution from failure tolerant research-oriented Grid systems into mission critical commercial ones.

1.2.2 Research Objective

The objective of this research is to introduce a novel reputation paradigm for managing resource selection in Grid computing environments [77, 78, 79]. This paradigm endeavours a synergistic *reputation-policy* based trust model which enables different Grid client applications (e.g. meta-schedulers [4, 5, 3], resource brokers [1, 2, 6], monitoring toolkits [86, 87, 88] and information systems [89]) to carry out an active participation in the trust and reputation evaluation process [77]. This is achieved by enabling these Grid clients to extend their existing reputation queries with a set of reputation-policy assertions rectified as trust decision strategies.

The motivation behind the reputation-policy trust model is driven from the fact that this research considers existing Grid reputation-based trust models to be rather limited as that they do not allow external involvement in the trust evaluation process. Currently, the Grid client applications are not able to calculate the trust value of a Grid resource by specifying their own trust evaluation criteria and as a result, they are obliged to rely on an esoteric community-based reputation algorithm to compute trust values. While this may provide an adequate coverage for preliminary resource evaluation scenarios, more advanced approaches should be considered, given the highly subjective and context dependent nature of trust and reputation.

In order to support the research objective, this research follows a problem-solving approach where the intention is to discover the range and the criteria (i.e. the boundaries) by which the hypothesis made regarding the reputation-policy trust model is applicable. This implies identifying a real world case study and performing an iterative comparison analysis against an existing reputation model to discover the circumstances by which the reputation-policy trust model has distinct advantage in filtering low quality resources as well as the circumstances in which it has not.

1.3 Trust Management Concepts & Challenges

1.3.1 Concepts

Definition 1: Trust is a complex concept which had been a subject of research in different fields including sociology, business, law and computing [11]. In computing literature, Marsh [11] was the first person to introduce *trust* in distributed artificial intelligence. In the context of this thesis, trust is based on Gambetta's [13] theoretical work, and envisions it as the subjective belief a trusting agent has in the

capability and willingness of a trusted agent to deliver a quality service or product in a given context and time slot. This belief is based on the trusting agent's direct and indirect experiences that had been formerly achieved with the trusted agent [10, 66].

A *trusting agent* is an entity who has faith or belief in another entity in a given context and at a given time slot whereas a *trusted agent* is an entity in whom faith has been placed by another entity in a given context and at a given time slot [10, 92].

Trust is realised by the concept of a *trust relationship*, which defines a *bond* or *association* between a trusting agent and a trusted agent [10]. The strength of a trust relationship between two agents can be represented numerically using a *trust value*. The trust value depicts the amount of trust in the trust relationship that the trusting agent has in the trusted agent in a given context and in a given time slot [10]. This value can be quantified to determine the *trustworthiness* of the trusted agent utilising a well-defined *trustworthiness scale* [10]. This scale provides a metric which measures the *trust level* the trusting agent has in the trusted agent. In the context of this thesis, trust level is regarded as benchmarking criteria which is intended for managing the *execution risk* of submitted jobs. Execution risk refers to an adverse effect on a running job as a result of a failure of one or more Grid resources. This may be the result of resources not being available at the time of the execution, inaccurate or even malicious results, slow response time and etc.

Definition 2: Reputation has been used in various disciplines like sociology, economics and psychology whilst in computing literature, the concept of reputation has been applied to multi-agent systems [10]. Reputation indicates the general perception of the trustworthiness of a trusted agent as perceived by peer agents [18]. It is a concept closely related to trust, as it is considered as a quantifiable measure of trustworthiness [52]. In the context of this thesis however, reputation is based on Abdul-Rahman and Hailes [31] theoretical work and envisions reputation as the aggregation of all the recommendations from third-party recommendation agents regarding the service quality of a trusted agent in a given context and time slot.

A *third-party recommendation agent* is an entity which provides a reputation feedback regarding the quality of a service, product or a trusted agent. [10].

A *reputation feedback* is a statement issued by a recommendation agent regarding the Quality of Service (QoS) provided by a trusted agent in a single transaction. Each reputation feedback is comprised of an opinion and recommendation value [10].

An *opinion* is a general impression of a recommendation agent regarding a trusted agent derived from its feedbacks on all the transactions that were conducted with the trusted agent [80]. In the context of this thesis, an opinion refers to an impression regarding a single quality aspect (i.e. availability, reliability and etc).

An aggregation of opinions produced by recommendation agents constitute as

the *reputation* of a trusted agent. It is expressed using a *reputation value*, which is an aggregated value that represents the total recommendations made by third-party recommendations agents regarding the trustworthiness of the trusted agent [10].

A *reputation query* is the inquiry made by a trusting agent for a specific context and time slot regarding a service, product or a trusted agent. The reputation query typically includes a context ID, context description, context time and etc [10].

A *context* is defined as the nature of a service or service functions (e.g. 'store data') or alternatively defined as an object or entity (e.g. 'file system') [10].

An important characteristic of reputation is that it is *multi-faceted*. This means that reputation is comprised of multiple aspects, such as availability, reliability, honesty and etc [92]. In the context of this thesis, this characteristic is maintained throughout both reputation queries, which support querying multiple quality aspects and decision criteria as well as reputation feedbacks, reflecting the recommendation agent evaluation on a variety of aspects of a service e.g. price, product and quality.

Figure 1.1 facilitates associating the relationship between trust and reputation. According to the figure, reputation is a recommendation which perceives trustworthiness. Trustworthiness is a metric which quantifies the strength of a trust relationship. A trust relationship is an association which realises trust whereas trust itself is a belief of a trusting agent in the capabilities and willingness of a trusted agent to deliver a quality service or product for a given context and timeslot.

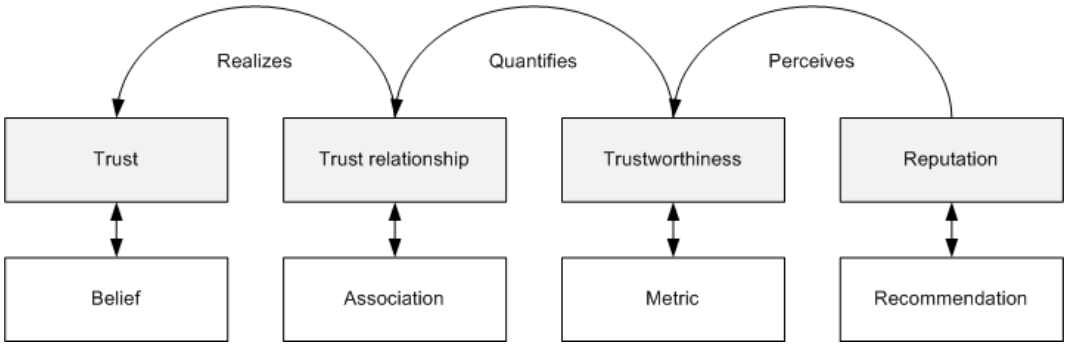


Figure 1.1: Trust and reputation building blocks

Definition 3: Policy is generally described as a statement of the intent of the owner or controller of some computing resources, specifying how he wants them to be used. Policy specification languages, such as [44, 45, 46] as well as the W3C WS-Policy specification [47] are an attempt to formalise the intent of the owner into a form that can be read and interpreted by machines [32]. A policy statement may give certain rights to entities (programs, users, communications, etc) that fulfil some criteria and deny other rights. Each language must define the entities and their attributes it considers as well as the actions or permissions that can be given.

However, in the context of this thesis, policy is regarded as a permutation of *reputation-policy assertions* (i.e. a set of reputation evaluation requirements) expressed by trusting agents which are used for interrogating the trustworthiness of trusted agents. These policy requirements are derived from the trusting agents intrinsic view on trust, the global context and the type of job they wish to submit.

Figure 1.2 illustrates the usage of policy in the context of this research. The trusting agent submits a Reputation-Policy Query (RPQ), which is essentially a reputation query fortified with reputation-policy assertions. This query instructs the recommendation agents regarding the reputation criteria for which to extract information for. The reputation-policy query contains instructions for aggregating and adjusting the feedbacks from the recommendation agents to the reputation evaluation requirements of the trusting agent. The query also contains decision instructions which map the aggregated feedback values into a trustworthiness scale which measures the level of trust that the trusting agent should have in the trusted agent. The third-party recommendation agent provides a reputation feedback (i.e. recommendation) by rating the quality of service supplied by the trusted agent.

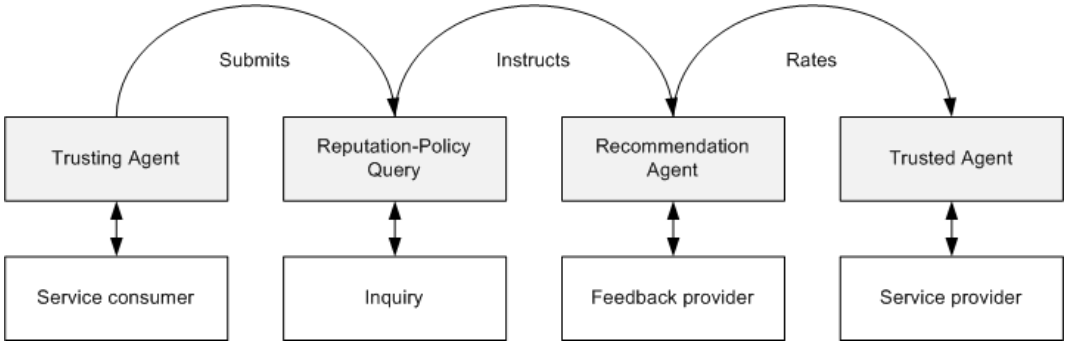


Figure 1.2: Reputation and policy building blocks

1.3.2 Challenges

There are several challenges imposed on a synergistic reputation-policy based trust model which are required to be addressed in order to introduce subjective capabilities into reputation management in Grid computing. The following sections describe the challenges and corresponding assumptions considered for designing the model.

Trust & reputation model is concerned with the arrangement of trust data. The first property of interest is whether the trust mechanism is centralised or decentralised. Centralised models have the disadvantage of a single-failure point. However, in classical Grid systems, where security is achieved through certificates and central certification authorities exist, a centralised model can be favourable as it can be seen

as an interrogation service storing records of historical ratings feedback data used for calculating the trustworthiness of disparate Grid resources [52].

In addition, the reputation-policy trust model is concerned with the creation, management and storage of trust decision strategies. Dynamic strategy creation requires explicit client knowledge regarding the nature of the submitted job combined with intrinsic information of the characteristics of the Virtual Organisation (VO) [8] and the depth of historical ratings feedback data. Since dynamic trust decision strategy creation is outside the scope of this research, it is assumed that whenever Grid client applications submit reputation queries, the attached strategies had been previously generated based on the client knowledge factors. Nevertheless, the secure storage and retrieval of the trust decision strategies remain an issue to be addressed by the synergistic reputation-policy based trust decision model model.

Trust and reputation metrics is considered as the value that expresses the trustworthiness of an entity that is provided by the reputation algorithm. Typically, these values are scaled between 0 and 1 (0 - no trust, 1 - maximum trust). The reputation-policy based trust model is challenged with both generating these trust values as well as mapping these *trust values* onto *trust levels* using a trustworthiness scale. These are exclusively based on the metrics supplied by the Grid client application, which provides both evaluation criteria as well as decision mapping rules. The main issue in concern is regarding the definition of a general purpose metrics model which would be able to accommodate different reputation requirements set by Grid client applications while adhering to well formed structure and semantics.

In particular, the model has to address the challenge of preserving reputation as a multi-faceted aspect of trust while extending its capabilities to support it's *subjective* nature. In turn, this challenge dictates that Grid clients should be able to derive their reputation evaluation criteria using inductive synthesis over the different aspects of trust. The synthesis is required to include the selection of one or more trust aspects, definition of the sources for each aspect (this is described in the following chapter) and controlling the importance of each aspect using weight factor rules.

An additional challenge is to model the *opportunistic* behaviour Grid clients may pursue in order to control trust based decisions considering the *uncertainty* nature of trust. In practical terms, this requires allowing rule-based mapping of trust values, which were formerly generated using the reputation evaluation criteria into trust levels which define thresholds for controlling Grid resource selection. The combination of the evaluation criteria with the rule-based decision mapping is required to produce the trust level values of the reputation-policy based trust model.

The consecutive challenge is to provide a Grid reputation-policy trust management architecture which provides a solid reference implementation for the reputation-

policy trust model (GREPTrust). As the proposed trust model is a novel introduction for the Grid computing environment, the architecture is challenged with adapting the model according to the Grid computing trust and security requirements while consistently adhering to the structure and definition of the model.

Reputation querying management is embraced with the type of data associated with the reputation querying mechanism as well as processing this data for generating trust values. Typically, a reputation query will include the identifier of the entity making the reputation query as well as the identifiers of the resources to be evaluated. The introduction of the synergistic reputation-policy based trust model imposes new challenges on the type of data required to be submitted along with the reputation query, as it allows a *heuristic* control over the behaviour of the reputation algorithm. For example, Grid client applications should be able to control the reputation algorithm by setting properties such as the cut-off time (COT) [10] for gathering the historical data, the trust decay function (TDF) [10] to be used for controlling historical precedence as well as the actual reputation-policy assertions.

The consequent challenge is focused on transforming the reputation evaluation criteria denoted by the trust decision strategy into trust value for each evaluated resource. The major issue in concern is concentrated on correlating each trust aspect specified inside the strategy and matching it with its historical rating feedback counterparts. This mechanism should operate on each aspect independently and then compute the total trust value once each correlation had been completed. There could be several issues that may interfere with achieving this correlation. For example, trust decision strategies may include trust aspects which are not supported by the VO and therefore violating the integrity of the reputation algorithm. In addition, strategies may include trust aspects which do not have corresponding historical feedbacks available and as a result the validity of computation may be impacted.

Finally, the produced trust values are required to be mapped into trust levels taking into account the decision rules supplied by the Grid client. This challenge should take into account the subjective perception on trust and uniform these uncertainties into a crisp value which will denote a threshold for resource selection.

Reputation feedback Grid clients are expected to rate the quality of the transactions they have been engaged with using an evaluation feedback mechanism. This mechanism accepts reputation feedback in the form of a continuous value from the Grid client in order to rate the transaction. This value is incorporated directly into the model's direct trust component. Reputation information is typically positive or negative and it represents delivered service feedback values. These feedback values are based on the formal contracts, service standards or service level agreements

(SLA) which had been agreed between the entities prior to the transaction.

The major challenge is focused on generating reputation feedback values from the discrepancies between the service contracts and the actual level of service delivered. Moreover, since the reputation-policy based trust model treats reputation as a multi-faceted, Grid clients are expected to rate each evaluated quality aspect separately so the challenge must be addressed by taking into account the trust decision strategy when performing the reputation feedback calculation. The reputation-policy based trust model assumes that all Grid clients submit feedbacks daily and none of the feedbacks are biased for or against any Grid resource.

1.4 Research Deliverables

1.4.1 Research Contributions

This research extends the frontier of current reputation-based trust management solutions applied to Grid computing environments. The contributions made by this research to scientific knowledge lies in the following points:

RC1. **Synergistic reputation-policy trust model:** The key contribution of this research is a synergistic reputation-policy trust model. This contribution promotes a trust model which allows Grid client applications (e.g. resource brokers, schedulers and monitoring toolkits) to control the trust and reputation evaluation process. The main artifacts of the trust model are: *Trust Decision Strategy* (TDS) - structured document which contains reputation-policies consisting of a finite set of opinions and decision mapping rules. *Opinion Matrices* (OM) - tabular data structures which store the historical ratings feedback values reported by trusting agents. *Correlation Process* (CP) - A matchmaking process which involves a reconciliation of each opinion element stipulated in the trust decision strategy with it's aggregated historical ratings in each opinion matrix counterpart thus, in order to compute trust values.

RC2. **Grid reputation-policy trust management system architecture:** The following contribution is to derive a service oriented architecture which accommodates the different artifacts of the reputation-policy trust model. It is comprised of three underlying domains: *client domain*, *service domain* and *data domain*. The client domain is concerned with storing and obtaining the trust decision strategies as well as constructing reputation queries. The service domain is concerned with providing means of querying resources based on the reputation-policies defined in the trust decision strategies as well as means for providing reputation feedbacks for rating transactions. The data domain is concerned with storing and manipulating historical reputation feedback data.

RC3. **Grid resource reputation querying mechanism:** The purpose of the reputation querying mechanism is to process the reputation evaluation criteria denoted by the trust decision strategy and calculate trust level values for a Grid resource or a set of resources. The processing occurs in three steps. The first step calculates trust values by processing the evaluation model and generating a trust value for each resource. The second step translates the trust value intro a trust level by processing the decision model. The third step generates a report and returns it to the Grid client.

RC4. **VO aggregated reputation querying mechanism:** Having defined the Grid resource reputation querying mechanism contribution, the consecutive contribution is to allow the reputation algorithm to support global trust context. This allows evaluating the trustworthiness of an entire VO based on the individual trust of its members. In addition, this feature allows the inference of the trust level of a single resource based on the trust of the global context.

Figure 1.3 illustrates the research contributions in the context of the reputation policy trust model. **RC1** introduces the model and its internal artifacts. **RC2** translates the model artifacts into an trust management architecture suitable for a Grid computing environment. This contribution introduces the architecture subsystems and components as well as underlying data model. **RC3** and **RC4** concentrate on the resource query management component for querying the reputation of a single resource, set of resources contributed by an organisation or even an entire VO.

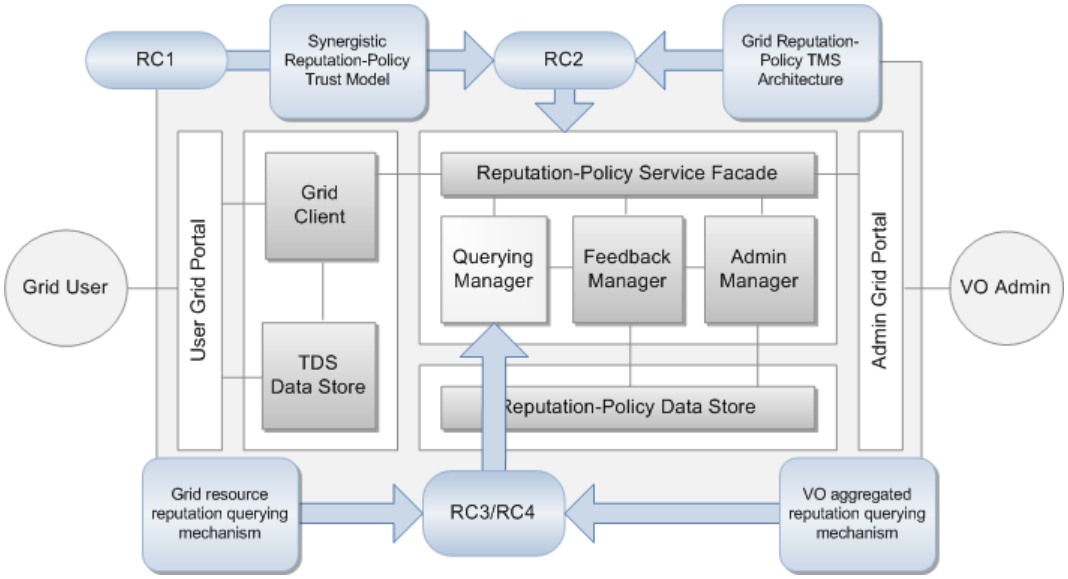


Figure 1.3: Synergistic reputation-policy trust model contributions

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

- *Chapter 2: Background Research, Related Work & Critical Analysis*
  This chapter reviews the state of art with regards to Grid reputation-based trust management systems. The objective of this chapter is to provide a solid rationale for introducing the synergistic reputation-policy based trust model. This is achieved by describing the merits and limitations of the existing solutions and introducing the aspects that the proposed model should consider.

- *Chapter 3: Synergistic Reputation-Policy Based Trust Model*
  This chapter describes the requirements and characteristics of the synergistic reputation-policy trust model and it's underlying artifacts (RC1).

- *Chapter 4: Grid Reputation-Policy Based TMS Architecture & Querying mechanism*
  This chapter presents the Grid Reputation-Policy Trust management system architecture (GREPTrust) (RC2). Being derived from the synergistic reputation-policy trust model, this chapter further elaborates on the trust data management as well as Grid resource and aggregated querying mechanisms as specified by the system architecture (RC3 & RC4 correspondingly).

- *Chapter 5: GREPTrust testbed, Case Study & Simulation Design*
  This chapter encompasses three sections. In the first section, it introduces the GREPTrust testbed infrastructure and describes it's requirements, underlying procedures as well as design constraints and resolutions. The second section presents a computational finance case study which provides tangible context for evaluating the reputation-policy trust model. The third section describes a workflow simulation based on the requirements presented by the case study.

- *Chapter 6: Experimental Results & Post-mortem Analysis*
  This chapter presents three clusters of experimental results produced by the GREPTrust testbed architecture and a post-mortem analysis based on these results. Based on the case study and the simulation design described in chapter 5, these results provide side-by-side comparisons of outputs generated by Grid computations relying on GridPP and GREPTrust resource recommendations.

- *Chapter 7: Conclusions*
  This chapter summarises and reflects on the research activities, highlights the contributions to knowledge produced throughout the course of this thesis and finally presents suggestions for future work. The main part of the thesis is

finalised with an alphabetical glossary used for clarifying the list of terms, definitions and abbreviations used throughout the chapters.

- Appendix 1 contains a dissemination of the research findings including a list of publications, conferences, workshops and awards.
- Appendix 2 contains UML diagrams of the GREPTrust architecture as well as examples of TDS, Reputation-Policy Report (RPR) and Monitoring and Discovery Service (MDS) files. In addition, it includes sample outputs of querying different aspects of the system (e.g. set of resources and VO).
- Appendix 3 contains UML diagrams of the GREPTrust testbed architecture as well as detailed design (in pseudo code) of the main Volume Distribution Profile (VDP) generation simulation algorithms and simulation outputs.
- Appendix 4 contains a dissection of the experimental results.

1.6 Conclusions

The objective of this chapter was to introduce the framework for the research. This chapter initiated by setting the background theme - i.e. formulating a discussion over the importance of trust and reputation management in various computing environments and in Grid computing in particular. Consecutively, it recognised the need for a robust reputation solution which optimises resource selection based on client evaluation criteria. The main outputs of this chapter are the following:

- Main research hypothesis proclaiming that a synergistic reputation-policy based trust model would allow an elevated degree of fine-grained resource selection. This is accompanied by:
  1. Problem statement including a generalisation of the current state-of-art and a rationale for considering an exoteric philosophy in trust & reputation management.
  2. Research objective aiming to introduce a novel synergistic reputation-policy based trust model and accompany it with an empricial proof based on comparison analysis.
- Description of four research contributions (RC1-RC4) which would extend the frontier of current reputation-based trust management solutions applied to Grid computing environments.

Chapter 2

Background Research, Related Work & Critical Analysis

"Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under thy observation in life."

Marcus Aurelius

2.1 Introduction

The purpose of this chapter is to provide a solid rationale for introducing the synergistic reputation-policy based trust model for improving Grid resource selection. It is therefore comprised of three sections: background research, related work and critical analysis. The background research section follows a narrow down approach where it initially discuses the concepts of trust management as defined in the academic literature. Consecutively, it reviews the methodologies commonly used for managing trust and finally it describes the key elements available in reputation-based trust management systems. The related work section includes a review of trust management from the Grid computing perspective as well as a survey of the state of the art reputation-based trust management systems available in Grid computing. The critical analysis section discusses the merits and limitations of each of the existing solutions, makes a generalisation regarding the overall limitations through an hypothetical user scenario and finally introduces the aspects that the reputation-policy trust model should consider in order to address these limitations.

2.2 Background Research

2.2.1 Trust Management

Trust is the fundamental aspect of any secure communication and a crucial ingredient in any mutual relationship and where transactions are carried out in a distributed environment to provide the agreed to the Quality of Service (QoS) [10]. Consecutively, trust management has been an important research contribution for the development of modern open distributed and decentralised systems. It has been studied in the context of decentralised access control [95, 42, 103], public key certification [96, 97] and reputation systems for peer-to-peer networks [43, 98, 99, 100].

Trust management was initially introduced by M. Blaze et al. [95] as:

> *"a unified approach to specifying and interpreting security policies, credentials, and relationships which allow direct authorisation of security-critical actions".*

The basic idea is that they acknowledge the incompleteness of security information in open systems and suggest that systematic security decision demands extra security information. Afterwards, different researchers have carried out thorough studies on trust modelling as well as trust management technologies from different aspects and in different environments [22, 94]. In particular, trust management has been given a broader definition by T. Grandison [94], which is not limited to merely authorisations:

> *"Trust management is the activity of collecting, encoding, analysing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships".*

This thesis adopts the definition made by T. Grandison regarding competence and therefore, the focus of this research is on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behaviour. The reason for this adoption is that in the context of this thesis, trust management is addressed in terms of assessing the *competence* of delivering Quality of Service (QoS) rather than providing access control authorisation. In addition, T. Grandison explains that the purpose of trust management is for making assessments and decisions regarding trust relationships. This accords with the purpose of the reputation-policy based trust model presented in this thesis.

In [21, 22], several trust classifications are made regarding the different types of trust management available in the academic literature. However, in the context of this thesis, a distinction is being made between two predominant classes of trust management: *identity* and *behaviour*. These classifications are based on the formal definitions made by Azzedin et al. [16]. Table 2.1 describes and compares the different types of trust management. Among the two classes, behaviour trust management is the focus of the work described in this thesis. The reason for placing the focus on behaviour trust is that it concentrates on the trustworthiness of systems and institutions that are in place in order to support the transaction and provide a safety net in case something goes wrong. This characteristic is of paramount importance for supporting dynamic and pervasive computing environments [52].

2.2.2 Trust Management Systems

Trust Management Systems (TMS) are responsible for managing trust relationships in a distributed environment [49]. They are mainly characterised by scalability, reliability, and security [80]. Management of trust within a trust management system includes: (i) negotiation of trust when a new member joins the distributed environment, (ii) storage of the trust metrics, and (iii) distribution of trust metrics [80].

Trust management systems support four activities [21] (illustrated in Figure 2.1); *Trust Evidence Collection*, which is the process of collecting evidence required to make a trust decision; *Trust Analysis*, which is the process of examining trust relationships to identify implicit relationships; *Trust Evaluation*, which evaluates evidence in the context of trust relationships (i.e. computes the trust value based on a predefined trust function) and *Trust Monitoring*, which is the activity that is responsible for updating trust relationship based on evidence. Therefore, building a trust management service for activity-aware trust applications requires fulfilling a set of requirements which can be categorised in groups that represent a TMS activity [33].

Types of Trust Management Systems

The academic literature has recognised the emergence of two types of trust management systems: *policy-based* and *reputation-based* [49, 80, 81]. These two methodologies have been developed within the context of different environments and targeting different requirements [37].

In *policy-based* trust management systems, the different entities or components constituting the system, exchange and manage credentials to establish the trust relationships which are further refined based on certain predefined policies. Their main goal is to enable access control by verifying the credentials (certification trust) and restricting access to credentials-based predefined policies (resource access trust) [49].

Table 2.1: Comparison of trust management types

#	Class	Type	Description
T1.	Identity	Certification	- Based on the certification of the trustworthiness of the trusted agent by a third party, so trust would be based on a criteria relating to the set of certificates presented by the trusted agent to the trusting agent. - Trust systems that derive certification trust are typically authentication schemes such as X.509 identity certificates and PGP [115].
T2.	Identity	Resource access	- Describes trust in principals for the purpose of accessing resources owned by the relying party. A trusting agent trusts a trusted agent to use resources that he own or controls. - Been the focus of security research for many decades [31], particularly on mechanisms supporting access control. - Forms the basis for specifying authorisation policies, which are implemented using access control mechanisms, firewall rules, etc [49] .
T3.	Behaviour	Context	- Describes the extent to which the relying party believes that the necessary systems and institutions are in place in order to support the transaction and provide a safety net in case something goes wrong. [49]. - 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 [66]. - Factors for this type of trust can for example be critical infrastructures, insurance, legal systems, law enforcement and stability of society in general [22].

This approach is closely related to identity trust and has been proposed in the context of open and distributed services architectures as a solution to the problem of authorisation and access control in open systems [39, 40, 41, 42, 103].

In *reputation-based* trust management systems, on the other hand, a mechanism is provided by which a system requesting a resource - resource consumer (RC) evaluates the trust of the system providing the resource - resource provider (RP) [80]. These types of systems are closely related to behaviour trust [49]. The trust values can be a function of the global and local reputation along with the different policies [80]. Key elements in this type of systems are the trust model, trust metrics

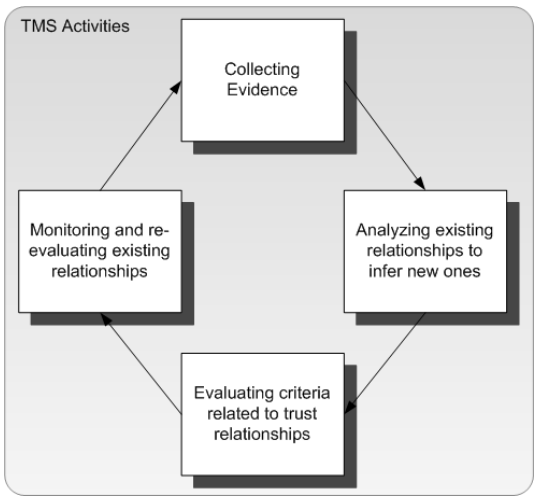


Figure 2.1: Activities supported by Trust Management Systems

and reputation feedback generation [49, 52]. Reputation-based trust management systems have emerged in the context of electronic markets, such as Amazon [34], eBay [35] and the iTunes store [36]. In distributed computing, reputation-based trust methodologies have been proposed for managing public key certificates, P2P systems [69, 70], mobile ad-hoc networks [71, 72] and in the Semantic Web [37].

In some later work, Bonatti et al. [38] proposed an integrated trust management framework that combines ruled-based and credential based trust (i.e. policy-based trust management) with numerical trust estimates based on a large number of sources (i.e. reputation-based trust management). This solution is capable of addressing the complexity and the variety of semantic web scenarios with both structured organisational environments and unstructured user communities.

Among the two types of trust management systems, reputation-based trust management is relevant to this thesis as the aim of this research is to optimise the quality of computations utilising the recommendations made by the presented system. Moreover, policy grammer is defined in terms of stipulating reputation evaluation requirements (as opposed to access control). Therefore, the remainder of this chapter will aim to focus on the key elements of reputation-based trust management systems and review the Grid reputation-based trust management systems.

2.2.3 Reputation Based Trust Management Systems

Trust Model

The trust model (also known as the computational model [52] or computational engine [22]) is a quantitative model for producing trust values (i.e. scores) from the interactions of agents or from the recommendations or history of agents.

The academic literature differentiates between various types of trust models based on the *information* used for producing trust values [22, 23, 66]. Trust values are typically computed based on a trusting agent personal experience (or private information) with a trusted agent depicting the amount of trust in the trust relationship that the trusting agent has in the trusted agent in a given context and time slot. This type of interaction is defined in the literature as a *direct trust* association.

Alternatively, trust values can be computed using a combination (i.e. blend) of personal experience with second hand referrals (also known as public information), which are essentially reputation feedback ratings provided by third-party recommendation agents regarding their personal experiences with the trusted agent. This is defined in the literature as an *indirect trust* (or reputation) association. In contrast, reputation values are solely based on indirect trust. Sabater et al. [83] make a classification of trust and reputation values based on several criteria, such as the conceptual model, information sources and etc. However, in the context of this thesis no distinction is being made regarding the terms *trust value* and *reputation value*. The reason for this this decision is that computed values produced by the presented trust model can be based on *sourcing* either direct trust or indirect trust or alternatively a combination of both direct and indirect trust. This also serves the advantage of simplifying the rather complex terminology.

Therefore, the term ***trustsource*** is used in this context to define an input value to a trust model. This input practically takes the form of time series data points containing either direct experience ($\{S_e\}$), indirect (or reputation) second hand referrals ($\{S_r\}$), a combination of the two series points ($\{S_e, S_r\}$) or an empty set ($\{\oslash\}$) - in case no input is supplied. Considering set $S = \{S_e, S_r\}$, the following defines ***trustsource*** s as unique element: $\exists! s \in \mathbb{P}(S)$. Based on the previously described, the following generalisation is made in this thesis regarding a typical dependency ($s \vdash v$):

$$\textit{trustsource} \overset{\textit{input}}{\rightarrow} \textit{trustmodel} \overset{\textit{output}}{\rightarrow} \textit{trustvalue} \tag{2.1}$$

where ***trustmodel*** includes within scope a *trust function* providing mappings between ***trustsource*** s and ***trustvalue*** v , such as $(f : s \rightarrow v)$. The distribution of the trust model is an important consideration when designing reputation-based trust management systems and advantages/disadvantages of centralised and decentralised trust models have been described by Silaghi et al. [52]. The following generalisation outlines the mappings ($v \vdash l$) to trust level l using a trustworthiness scale:

$$\textit{trustvalue} \overset{\textit{input}}{\rightarrow} \textit{trustworthinessscale} \overset{\textit{output}}{\rightarrow} \textit{trustlevel} \tag{2.2}$$

An important distinction is made in this thesis between *esoteric* trust models and *exoteric* trust models. Esoteric trust models are based on a confined trust function in which the computation of a trust value from a trust source is abstracted and encapsulated from the trusting agent. In contrast, exoteric trust models range between parametrised trust functions which allow certain aspects of the computation to be adjusted externally (e.g. weight ratios between direct and indirect sources) to complete exoteric models in which the trusting agent provides the trust function.

Trust Metrics

Metrics is defined as system of measurement, i.e. a system of related measures that facilitates the quantification of some particular characteristic. In the context of this thesis, trust metrics refers to the actual value that expresses the trust (or trustworthiness) of an entity provided by the trust mechanism [52]. The importance of trust metrics had been highlighted by Alunkal et al. [57] where they recognised trust metrics as a building block to support Quality of Service (QoS) requirements. In contrast, Jøsang et al. [22] reflected on trust metrics from *reliability* perspective where they described the different types of supported metrics, such as discrete, continuous and normalised values. Continuous metrics are considered more expressive than discrete ones. However, they require more cognitive effort in order to be interpreted efficiently by humans [23]. Usually, trust values are scaled between -1 and 1, or between 0 and 1. If the reputation scheme uses values scaled between 0 and 1 these values can have the meaning of a probability [52].

In the context of this thesis, *normalised-continuous* trust metrics are utilised for expressing both trust values as well as trust levels. This is justified by the fact that the trust metrics produced by the trust model presented in this thesis are intended to be interpreted by machines (rather than humans) - hence the decision on selecting continuous values rather than discrete ones. In addition, the presented trust model aspires to provide a subjective measure of probability - hence the decision on scaling trust values v and trust levels l between 0 and 1 (i.e. - $v, l \in [0, 1]$).

Reputation Feedback

An in depth description of different types of reputation feedbacks is available in [67] and includes: simple feedback, weighted feedback, beta filtering feedback, beta deviation feedback and selective weighted feedback. Typically, reputation information might be positive or negative one. Some systems are based on collecting both type of information with regard to an entity [35], while other systems are based only on negative/positive information [14]. The following generalisation extends formula 2.1 outlining the relationship between reputation feedback and trust source:

$$recommendationagent \overset{output}{\rightarrow} reputationfeedback \overset{input}{\rightarrow} trustsource \tag{2.3}$$

Regarding an accomplished transaction, the recommendation agent can supply with either binary, discrete or continuous values. Similar to trust metrics, continuous values are more expressive but for the sake of simplicity, numerous approaches use discrete feedback and later on aggregate this feedback in continuous reputation or trust. Concepts from utility computing could be used to generate feedback on transactions as presented by Silaghi et al. [53]. Three important challenges in reputation feedback management are identified in the academic literature as: (a) low incentive for providing ratings, (b) bias towards positive ratings and (c) unfair ratings [22].

Trust Evaluation Methodologies

The metrics for trust is driven from the trust mechanism (or methodology) that produces it. As a result, different trust and security research initiatives produced several trust evaluation methodologies. These were comprehensively reviewed by Jøsang et al. [22] who analysed computational engines classified in accordance with their category: simple summation, Bayesian systems, discrete trust models, belief models, fuzzy models and flow models.

Brinkløv et al. [66] describe common methodologies used by trust models that combine direct trust and reputation as well as models the solely concentrate on reputation. A different approach for classifying trust evaluation methodologies is proposed by Zhang et al. [65] where trust functions are distributed in different categories: (a) subjective *versus* objective, (b) transaction based *versus* opinion based, (c) complete *versus* localised information and (d) rank based *versus* threshold based [80].

In the context of this thesis, trust methodologies are not classified by the type of trust metrics they produce but rather by the philosophical approach they pursue based on the notion of trust. This method of organisation facilitates the comparison of Grid reputation-based trust management systems in the following sections. Therefore, three approaches are classified as follows: *deterministic* approach, *probabilistic* approach and *approximative* approach. Each approach embraces one or more trust methodologies selected as relevant for the work described in this research.

Definition: Deterministic Category. The deterministic category is based on a philosophical view that envisions every event, including human cognition, behaviour, decision, and action, as causally determined by prior events [130]. It represents a

predictable doctrine, which does not involve any stochastic calculations for computing trust values. This category includes the *simple summation*, the *Average of Reputation Ratings (ARR)*, the *discrete* models and the *utility* models.

Simple summation [25], is considered as the simplest form of computing reputation scores where the principle is simply to sum the number of positive ratings and negative ratings separately for trusted agent y, and to keep a total score as the positive score minus the negative score:

$$\underbrace{\Gamma(y)}_{Trust} = \underbrace{\sum_{k=1}^{n} P_r(y)}_{Positive} - \underbrace{\sum_{k=1}^{n} N_r(y)}_{Negative} \tag{2.4}$$

ARR is slightly more advanced scheme proposed in [26] is to compute the reputation score as the average of all ratings, and this principle is used in numerous commercial web sites [34] as well as in the GridPP collaboration [133]:

$$\underbrace{\Gamma(y)}_{Trust} = \frac{\sum_{k=1}^{n} P_r(y)}{n} \tag{2.5}$$

Advanced methodologies in this category, such as the *discrete* models, compute a weighted average of all the ratings, where the rating weight can be determined by factors such as rater trustworthiness/reputation, age of the rating, distance between rating and current score etc. This type of models has been particularly promoted by Azzedin and Maheswaran [14, 15, 16], who used linear combinations of direct and indirect trust functions to evaluate the trust $\Gamma(x,y,t,c)$ of trusting agent x in trusted agent y at time t and context c:

$$\underbrace{\Gamma(x,y,t,c)}_{Trust} = \alpha * \underbrace{\Theta(x,y,t,c)}_{Direct} + \beta * \underbrace{\Omega(y,t,c)}_{Indirect} \tag{2.6}$$

where α and β $(= 1-\alpha)$ coefficients are weight factors. Θ is the direct trust function, computed as the product of the trust value in the *direct-trust table (DTT)* and the *trust decay function (TDF)* representing the rate at which trust depreciates. It is expressed as $\Upsilon(t-t_{xy},c)$ where c is the context, t is the current time and t_{xy} is the time of the last transaction between x and y:

$$\underbrace{\Theta(x,y,t,c)}_{Direct} = \underbrace{DTT(x,y,c)}_{DTT} * \underbrace{\Upsilon(t-t_{xy},c)}_{TDF} \tag{2.7}$$

Ω is the indirect (or reputation) function, calculated as the average of the product of the trust value in the *reputation-trust table (RTT)*, the *trust decay function* $\Upsilon(t-t_{xy},c)$ and the *recommender trust factor (RTF)* $(R(z,y) \in [0,1])$ representing

confidence that there is no collusion between z and y:

$$\underbrace{\Omega(y,t,c)}_{Indirect} = \frac{\sum_{k=1}^{n} \overbrace{RTT(z,y,c)}^{RTT} * \overbrace{R(z,y)}^{RTF} * \overbrace{\Upsilon(t-t_{xy},c)}^{TDF}}{\sum_{k=1}^{n}(z)} \tag{2.8}$$

where z is the domain of entities excluding trusting agent x, such that $\forall z \neq x$. The factors considered discrete models are: (a) the trust decay factor; (b) the credibility of the third-party recommendation agent; and (c) the combination of direct and indirect trust functions including their weight factors. Azzedin and Maheswaran have presented several variations on this basic model, including for example variable contexts and the use of brokers [17], and have been evaluated their performance in the context of Grid computing.

Utility models represent the latest advances in this category. They are primarily promoted by Silaghi et al. [53] and are based on concepts from utility computing where the reputation of a trusted agent is produced via the aggregation of the reputation of each of its *issues of interest* (i.e. quality aspects) within the VO. In this type of models, users feedback is represented as a utility function which reflects the satisfaction a user perceives from consuming a service.

Definition: Probabilistic Category. The probabilistic category is based on a philosophical view that expresses knowledge or belief that an event will occur or has occurred. The advantage of probabilistic models is conveyed by the rich body of probabilistic methods that can be directly applied. This provides a great variety of possible derivation methods, from simple models based on probability calculus to models using advanced statistical methods [22]. In the context of this thesis, this category includes the *Bayesian statistics* and the *flow* models.

An overview of the Bayesian statistics methodology is provided in [32]. At its essence, the Bayesian statistics methodology utilises the well-known Bayes formula to determine the conditional probability $p(H \mid E)$ based on an occurrence of event H given symptom E. For example, H could represent a satisfactory occurrence of a service whereas E could represent the delivery history of that particular service. The conditional probability of H given E is given by the expression:

$$p(H \mid E) = \frac{p(H\&E)}{p(E)} \tag{2.9}$$

The probability $p(H \mid E)$ is given by the probability of the occurrence of both H and E divided by the probability of the occurrence of E. Several researchers have used

the Bayesian methodology for modelling trust and reputation. The most notable of these are L.Mui [19], who used it for developing an expression for trustworthiness and Wang & Vassileva [20], who used it for determining reputation in P2P networks. In [91], a state of the art review is conducted in order to compare the existing Bayesian-network based methodologies for trust and reputation computation.

Flow models compute trust by transitive iteration through looped or arbitrarily long chains [22]. Some flow models assume a constant trust/reputation weight for the whole community, and this weight can be distributed among the members of the community. Participants can only increase their trust/reputation at the cost of others. The EigenTrust algorithm [31] is an example for a flow model. This algorithm computes agent trust scores in P2P networks through repeated and iterative multiplication and aggregation of trust scores along transitive chains until the trust scores for all agent members of the P2P community converge to stable values.

Definition: Approximative Category. The approximative category is defined as an estimation or inexact representation of information that is still close enough to be considered useful. It accounts as an adequate apparatus for dealing with the lack of precision inherent when describing trust and reputation [66]. In the context of this thesis, it includes the *fuzzy logic* methodology.

The fuzzy logic methodology perceives trust as a linguistic concept which is poorly described by simple numerical values, e.g. between -1 (total distrust) to +1 (complete trust). Based on the fuzzy set theory proposed by L. Zadeh [62], 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 [63].

There are two broad categories of fuzzy reasoning systems: The Mamdani method [73] and the Takagi & Sugeno method [74]. The basic idea is that for each behavioural parameter, a set of (possibly overlapping) *fuzzy sets* is defined in terms of *membership functions*, which specify the degree of membership of the various sets for the possible values of the parameter [62, 64]. For example, in Figure 2.2 three membership functions are defined (S_1, S_2, S_3). The Y-axis represents degree of membership ($\mu(x_1) \in [0,1]$) while the X-axis represents the input space for parameter ($x_1 \in [0,1]$). if x_1 has value 0.4, that it has degree of membership 0.25 in set S_1, 0.6 in set S_2 and 0.0 in set S_3. When the result of using several parameters $(x_1, x_2, \ldots, x_n)$ had to be combined to a crisp result y, a set of rules is used of the form:

$$\textbf{if } (x_1 \in S_1 \wedge x_2 \in T_j \wedge \ldots) \textbf{ then } y \in O_k \tag{2.10}$$

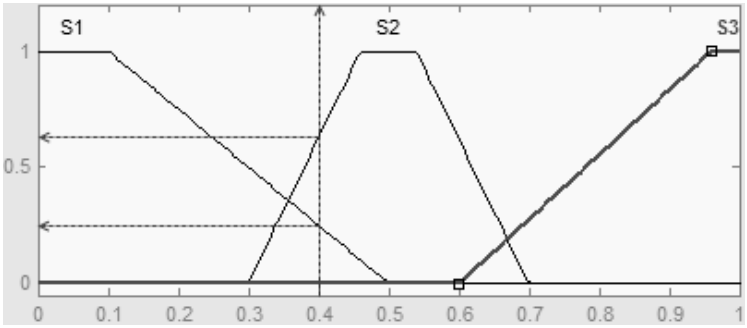


Figure 2.2: Fuzzy sets and membership functions

where $S_i, T_j, \ldots, O_k$ are all fuzzy sets and $\in$ indicates (some) degree of *fuzzy* membership. The membership function μ_k for the output fuzzy sets $O_k, k = 1, \ldots, r$ are very commonly given by the min-max formula:

$$\mu_{Ok}(y) = \max[\min[\mu_{Sk}(x_1), \mu_{Tk}(x_2), \ldots]], \; k = 1, 2, \ldots, r \tag{2.11}$$

For a given set of input parameter values $(x_1, x_2, \ldots, x_n)$, this gives a fuzzy output set. The actual output value, $y*$ is found from the membership function of $\mu_{Ok}(y)$ using a *defuzzification* method, such as *Centre of Gravity* (CoG) or *Mean of Maximum* (MoM) [82]. The use of fuzzy logic for trust evaluation has proven successful in various computing environments [59, 60, 61]. For example, the REGRET [84] system proposed by Sabater & Sierra uses fuzzy inference in order to describe to what degree an agent can be described as e.g. trustworthy or not trustworthy. It is important to realise that trust evaluation methodologies can be combined in a *synergistic* manner. For example, Brinkløv et al. [66] performed trust evaluation experiments based on combined methodologies, where reputation and trust evaluation where achieved using different methodologies. For example, *Fuzzy-Discrete* denotes fuzzy logic combination of trust where the reputation is calculated using the discrete reputation system and similarly for *Discrete-Discrete* and *Fuzzy-Beta*. Table 2.2 summarises the listed categories and associated trust evaluation methodologies:

#	Category	Methodologies
C1.	Deterministic	Simple summation ARR Discrete Models Utility Models
C2.	Probabilistic	Bayesian statistics Flow models
C3.	Approximative	Fuzzy logic

Table 2.2: Categories of trust evaluation methodologies

2.3 Related Work

2.3.1 Overview

The Grid security research has initially addressed trust through security mechanisms, such as GSI [55]. 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 Public Key Infrastructure (PKI), which defined message formats and protocols that allowed entities to securely communicate claims and statements as well as X.509 identity certificates, which entitled the nodes belonging to the trusted organisations, to join the Grid [110].

As a result, trust creation and management has involved human-coordinated process. When a new organisation wished to join the Grid, it had to fulfil the requirements set by the Certification Authority (CA) and wait for its approval before it was able to participate in the computation. Once the organisation was approved by the certification authority, it was considered trustworthy by the other resources.

This form of *certification trust* was satisfactory for static Grids where the set of users and resources remained constant during computations [55]. Some of the most notable initiatives and technologies at the time to support this type of trust management included MyProxy [107], the Liberty Alliance project [108] and Shibboleth [109], which offered solutions in credential management and identity federation.

With the expansion of VOs and the emergence of production Grid environments such as EGEE [121] and UK NGS [125], a requirement has emerged for managing trust in terms of authorisation and access control mechanisms. One of the earliest attempts at providing authorisation in VOs was in the form of the Globus Toolkit Gridmap file. This was later replaced by The Community Authorisation Service (CAS) [105], which was the next attempt by the Globus team to improve upon the manageability of user authorisation. Afterwards, the EU DataGrid and DataTAG projects developed the Virtual Organisation Membership Service (VOMS) [106].

VOMS has gone through a number of iterations during its development cycles. In one of its later incarnations it was integrated with a policy system (G-PBox) [50]. G-PBox is a policy handling service based on the XACML [129] specification which is designed to manage resource allocation and sharing. This integration made VOMS capable of defining and evaluating policies based on group and role membership and essentially transformed it into a *policy-based* trust management system.

This implementation of *resource access* trust addressed complex authorisation problems involving multiple administrative domains and multiple stakeholders [50]. Other authorisation systems such as Akenti [112] and PERMIS [111] employed policy mechanisms as well in order to manage access control in similar manner to VOMS.

However, as the Grid shifted toward ubiquitous and pervasive computing models, there became an increasing demand to be able to dynamically establish *behavioural trust* relationships between all entities once they joined the Grid. This is supported by Silaghi at al. [52] who expressed *reputation-based* trust management as one of the tools that the research community needs to supply in order to let the Grid expand beyond organisational boundaries. Arguably, reputation mechanisms are promising to perform well in Grid computing environments as they can bring with more dependability by addressing the sabotage-tolerance problem in *desktop* grids or by improving the resource allocation and scheduling in *classical* grids [52].

In the context of classical Grid environments, reputation-based trust management systems can be expected to improve reliability as well as the trustworthiness of resource selection thus further promote efficient resource sharing and the continuous evolution of the Grid [59]. This stands in accordance with the properties required for reputation-based trust models listed by Silaghi et al. [52]. They highlighted two properties as of particular importance to Grid computing: (a) SLA or QoS negotiation; and (b) trust aggregation. These two properties are highly relevant to the work described in this thesis as it targets QoS aware resource selection in classical Grid environments including trust aggregation on organisational and VO basis as well inference of a resource trustworthiness based on the global context.

Despite the acknowledgement over the importance of trust management in Grid computing [51], there are very few approaches for reputation models applied to classical Grid environments. Existing Grid reputation-based 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 [14].

As VOs are considered central in Grid computing, it is important to understand the role of a trust management system within a virtual organisation. Therefore, Table 2.3 provides an in depth analysis regarding the duties of a trust management system within the VO life cycle. This analysis is based on the requirements gathered for the TrustCom [113, 114] project regarding trust and security mechanisms. It facilitates conceptualising the following section, which provides a survey regarding the state of the art of Grid reputation-based trust management systems.

#	VO Phase	TMS Activity
V1.	Identification	**TMS selection**: Selection of the trust management system to be used (policy-based or reputation-based) and the trust policies that will be implemented.
V2.	Formation	**Trust negotiation**: The process by which all trust information - credential, reputation metrics, policies- is negotiated between the VO manager and the VO members.
V3.	Operation	**Resource monitoring:** This will be used as evidence when constructing the reputation of the resource providers. Unusual behaviours may lead to both a trust re-assessment and a contract adaptation.
V4.	Evolution	**Continuous restructuring:** Dynamically change the VO structure and replace resources which performed poorly or behaved inappropriately. This involves identifying new/alternative providers and resources as well as re-negotiating terms and providing configuration information as during identification, respectively formation phase.
V5.	Dissolution	**Resource assessment:** Assessment of the respective resources performances, e.g. amount of SLA violations and reputation. This may be of particular interest for further interactions respectively for other potential customers.

Table 2.3: Trust management in respect to VO life cycle

2.3.2 Literature Review

This section presents a comprehensive literature review regarding the different reputation-based trust management systems available for classical Grid environments. It is primarily based on the principles described in the background research, such as the trust model, the metrics for trust and reputation, reputation feedback and the categorisation of trust evaluation methodologies. The solutions presented in this section have been carefully selected to widely represent the state of the art in this area. They are summarised in Table 2.4 which highlights their key differences.

Each solution item is described as follows:

- **Classification:** The type of trust evaluation methodology & model.
- **Description:** Description of the solution.
- **Focus points:** Highlights key achievements/strengths of the solution.
- **Limitations:** Areas neglected by the solution or identified weaknesses.
- **Publications:** List of publications describing the solution.

GridPP

Classification: Deterministic » Average of Reputation Ratings (ARR).

Description: The Grid for Particle Physics is a British project for a particle physics Grid. It is part of the EGEE [121] project and it is the UK's contribution to the Worldwide LHC Computing Grid (WLCG) [122]. The GridPP reputation algorithm is internal to the project and is tightly integrated with the Service Availability Monitoring (SAM) system which is the basic monitoring tool for the LCG Grid. This monitoring tool regularly submits Grid jobs to sites and connects with a number of sensors in order to probe sites and publish the results to an Oracle database.

Historical *availability* and *reliability* ratings are maintained separately in the database regarding the contributed resources from the different institutions. Trust values are calculated as the 24hr (or 7 days) average of reputation ratings. The trust metrics can be represented using a discrete method: $trustvalue > 90\%$ implies full trust, $trustvalue > 75\%$ implies limited trust and otherwise non trust. If a resource fails certain tests (e.g. SAM tests), then groups of users can decide not to submit jobs to that resource. They do this by using the Freedom of Choice for Resources (FCR) tool which allows preemptive resource selection. The resource brokers then exclude sites that have failed the tests from jobs submitted by that VO.

Focus points:

- Tight integration with real-time monitoring and resource brokers.
- Straight forward, deterministic approach based on 24hr (or 7 day) ARR.
- Supports trust aggregation based on multiple level compartments (resource, organisation and VO) as well as multiple contexts (e.g. service delivery).
- Solution is available in a production Grid environment.

Limitations:

- Trust is practically regarded as single aspect since consolidation between availability and reliability is not available.
- No consideration for direct trust. The model solely relies on reputation ratings.
- Esoteric trust model. There is absolutely no support for controlling the trust calculation externally based on adaptation for specific job requirements.

Publications: [133, 134, 135]

Azzedin & Maheswaran

Classification: Deterministic » Discrete model.

Description: Azzedin and Maheswaran presented a formal definition of trust and proposed a model for incorporating trust to resource management systems in Grid computing. In their model, the amount of trust in which one entity has for another is a function of the direct experience between the two entities and their respective reputations. By differentiating between direct experience and reputation, the authors allowed different entities to weight these two components as desired.

In addition, they included a Trust Decay Function (TDF) in order to give less importance (weight) to old recommender ratings as well as a Recommender Trust Factor (RTF) which assigns lower weight to ratings made by business allies or previously engaged partners. Trust metrics are based on a simple discrete function ranging between $A-E$, such as A indicates a *very low trust level* whereas E indicates a *very high trust level*. Simulations were performed to evaluate the performance of a resource management algorithm that is trust aware against an algorithm that is trust unaware. Their simulation results indicated that the overall performance increased when the resource management algorithm was trust aware.

Focus points:

- General purpose solution for incorporating trust to resource management systems in Grid computing.
- Trust function uses weighted linear combination of direct and indirect trust.
- Inclusion of a Trust Decay Function (TDF).
- Inclusion of a Recommender Trust Factor (RTF) for minimising biased ratings.

Limitations:

- Single level compartment and limited number of contexts. There is no support for aggregating trust based on an organisational and VO basis.
- Trust is regarded as single aspect. There is no support for addressing multiple quality factors, such as availability, reliability, accuracy and etc.
- Virtually esoteric trust model; only weight factors for direct and indirect trust are allowed to be controlled externally.

Publications: [14, 15, 16, 17]

Aziz & Silaghi

Classification: Deterministic » Utility model.

Description: Aziz & Silaghi propose a reputation model for Grid-based virtual organisations that can be used to rate users as well as resources and their providers. This model is based on utility computing concepts, which expresses the satisfaction of an entity in its interactions with other entities with respect to some issues of interest. In addition, this model can be used to rate users according to their resource usage and resource providers according to the quality of service provided.

For each service, the user defines issues of interest and expected values on such issues. The satisfaction of the user over a service is measured by a *utility function*. Reputation of a service issue is then built by comparing its expected value with the actual value, delivered by a monitoring system. Reputation of a service is built as the aggregation of the reputation of its issues. Likewise, reputation of a service provider is built as the aggregation of the reputation of all services it delivers. This reputation model does not depend on direct feedback collected after the transaction. Experiments demonstrated improving the efficiency of resource brokering in Grids when using a reputation-based scheduling scheme. The model constituted as the basis for a design of a reputation-based trust management system that has been implemented in the EU FP6 project GridTrust [117].

Focus points:

- General purpose solution for integrating trust based on utility computing concepts where user feedback is represented as a utility function which reflects the satisfaction a user perceives from consuming a service.
- Tight integration with continuous monitoring information for building the reputation. This is has great advantage over post-transaction feedback.

Limitations:

- Users are obliged to provide the utility function together with the submitted job resulting in an increased message overhead.
- Esoteric trust model. Although the feedback mechanism is exoteric (using utility functions), there is no fine-grained control over the computation of trust value of an entity in a VO as it is predeterminately computed as the aggregation of the reputation of each of its issues of interest.

Publications: [53, 54]

GridEigenTrust

Classification: Probabilistic » Flow model.

Description: GridEigenTrust is based on the M.Sc. dissertation of B. E. Alunkal, Laszewski et al. [28] by exploiting the beneficial properties of EigenTrust algorithm [29]. GridEigenTrust extends the existing EigenTrust model to allow its usage in Grid computing. Trust management is integrated 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.

GridEigenTrust adopts a hierarchical model where reputation calculation is performed at three different levels: VO, institution and entity. An Entity's reputation is calculated as weighted average of new and old reputation values. An institution's reputation is calculated as weighted average of all underlying institutions reputation values. A VO's reputation is calculated as weighted average of all underlying institutions reputation values. Trust metrics are normalised as to scale to $[0, 1]$.

The reliability trust of an organisation could be obtained by a normalised weighted sum of the direct experience and the global trust in that organisation. This weighted sum is also added with the grade that users from trusting organisation assign to entities part of the trusted organisation. These global reliability trust values are used as normalised trust values in the EigenTrust model, being therefore, used to compute by iteration the global trust vector of the virtual organisation.

Focus points:

- General purpose solution for integrating trust as part of QoS management frameworks in Grid computing. It is based on the established EigenTrust model which demonstrated good performance P2P environments.
- Supports trust aggregation based on multiple level compartments (resource, organisation and VO) as well as multiple contexts (e.g. service delivery).

Limitations:

- Trust is regarded as single aspect. There is no support for addressing multiple quality factors, such as availability, reliability, accuracy and etc.
- Virtually esoteric trust model; only weight factors for direct and indirect trust are allowed to be controlled externally.

Publications: [28, 57, 30]

PathTrust

Classification: Probabilistic » Flow model.

Description: PathTrust is a reputation system proposed for member selection during the formation phase of the VO. When inviting members to join a VO, the initiator selects only those members whose reputation is above a certain threshold and probabilistically selects a member to be in the VO.

The reputation is built by aggregating positive and negative feedback the user submits after transaction execution. It arranges the participants in a graph structure where each edge in the graph is weighted with the trust between the nodes at the ends of the edge. The trust metrics is calculated by accounting the number of positive feedbacks let by participant i for participant j and subtracting the number of negative feedbacks weighted by the report between the total positive feedback and the total negative feedback participant i has submitted.

PathTrust was one of the first attempts to apply reputation methods to Grid computing by approaching VO management phases. They approached only partner selection and did not tackle organisational aspects. Their model still lacks of dynamics, as the feedback is collected only at the dissolution of the VO. But, the advance in the field is given by the fact that ideas from previous research were successfully transferred in the area of virtual organisations and computational grids.

Focus points:

- General purpose solution for member selection during the VO formation phase.
- Advance in the field is given by the fact that ideas from previous research were successfully transferred in the area of virtual organisations and grids.

Limitations:

- Trust is regarded as single aspect. There is no support for addressing multiple quality factors, such as availability, reliability, accuracy and etc.
- Approaches only partner selection and does not address organisational aspects.
- Trust model lacks dynamics. Feedback is collected only at the VO dissolution.
- Esoteric trust model. There is absolutely no support for controlling the trust calculation externally based on adaptation for specific job requirements.

Publications: [58]

SeGO

Classification: Approximative » fuzzy-logic.

Description: The Secure Grid Outsourcing (SeGO) system conceptualised and developed at the University of Southern California, is intended for secure scheduling of a large number of autonomous and indivisible jobs to Grid sites. The SeGO system has a different focus from the work described in this thesis, as it is focused on sabotage-tolerance by combining evaluations of *job success rate* (i.e. reputation) and *self-defence capability* to find an overall trust value while this thesis is focused on improving resource allocation and scheduling. However, a unique feature of the SeGO system is that its authors employ *fuzzy inference* for binding security in trusted Grid computing environments. The utilisation of fuzzy inference for deriving trustworthiness makes this highly relevant to this thesis.

As aforementioned, the SeGO authors define the trust metrics based on the *site reputation* and *self-defence capability* of a resource. The first characteristic is the behaviour attribute of a site and is composed of four parameters related to jobs behaviours like: Prior job execution success rate, cumulative site utilisation, job turnaround time, and job slowdown ratio. The second criterion is based on defence capabilities of a site namely, the IVS related capabilities, Anti virus capabilities, firewall capabilities, and secure job execution capabilities. Fuzzy logic is used to integrate the different capabilities to finally develop the trust metric.

Focus points:

- General purpose solution based on an hierarchical fuzzy-logic trust model specifically designed for distributed security enforcement in grids.
- Trust is regarded as multiple aspect as it is based on multiple quality factors.

Limitations:

- SeGO does not look at trust negotiation as an assumption is made that all resources have prior agreement for participating in Grid operations.
- SeGO makes an assumption that sites report their information honestly. Therefore, the possibility of malicious resources is not considered.
- Esoteric trust model as the permutation of quality factors is fixed as well as the fuzzy membership functions.

Publications: [60, 61, 68, 75]

Jia & Qu

Classification: Approximative » fuzzy-logic.

Description: Jia & Qu propose a reputation system for resource selection in computational grids, which is intended to leverage the guarantee of trustworthiness and reliability. Their reputation model offers both evaluation and decision making based on *fuzzy partial ordering* techniques thus in order to model the *multi-faceted* and *uncertain* features of trust. The proposed approach makes fuzzy partial order modelling on each resource provider's multi-faceted reputation, integrates overall information, and choose proper resource providers according to the final integrative order. The trust model is implemented on top of SOA, which has two fundamental components: Grid Reputation Service and Grid Contract Service offering a combination of reputation ratings and types of SLA negotiation contracts.

The main steps for resource selection are: first retrieve ratings for the candidate providers, then analyse and aggregate each aspect of ratings, finally according to the aggregated ratings from all aspects build fuzzy partial order relation and make the final selection.

Focus points:

- General purpose solution for combining resource provider selection with reputation mechanisms taking both QoS (Quality of Service) and QoP (Quality of Protection - i.e. sabotage tolerance) into unified consideration.
- The trust model is based on fuzzy partial order relations to model resource providers inferior and superior relationship. In addition, it supports service negotiation based on three types of SLAs: *Resource*, *Task* and *Binding*.
- Trust is regarded as multi-faceted. There is support for addressing multiple quality factors, such as availability, reliability, accuracy and etc.

Limitations:

- Esoteric trust model. There is absolutely no support for controlling the trust calculation externally based on adaptation for specific job requirements.
- There is no support for aggregating trust based on an organisational and VO basis as their solution merely addresses reputation of resource providers.

Publications: [59]

#	System	Classification	Centralised data	Trust metrics	Trust aggregation	Multi QoS aspects	Esoteric reputation querying	QoS/SLA negotiation	Type of feedback
1.	GridPP	Deterministic » ARR	yes	[0,1] or discrete	yes	yes (finite set)	yes	no	continuous
2.	Azzedin & Maheswaran	Deterministic » Discrete model	yes	discrete	no	no	yes	no	discrete
3.	Aziz & Silaghi	Deterministic » Utility model	yes	[0,1]	yes	yes	yes	SLA oriented	utility function
4.	GridEigenTrust	Probabilistic » Flow model	yes	[0,1]	yes	no	yes	QoS verification	continuous
5.	PathTrust	Probabilistic » Flow model	yes	[0,1]	no	no	yes	n.a.	negative & positive
6.	SeGO	Approximative » Fuzzy logic	no	[0,1]	yes	yes	yes	n.a.	negative & positive
7.	Jia & Qu	Approximative » Fuzzy logic	yes	[0,1]	no	yes	yes	SLA oriented	n.a.

Table 2.4: Summary of comparison between Grid reputation-based trust management systems

2.4 Critical Analysis

2.4.1 Discussion

As aforementioned, there are numerous trust management systems available as a result of countless research initiatives in various computing environments, such as the Internet, electronic markets, mobile ad-hoc networks and P2P systems. Reputation-based trust management is a rapidly growing research area in Grid computing offering solutions for the sabotage-tolerance problem in desktop grids as well as solutions for improving the resource allocation and scheduling in classical grids.

The state of the art solutions presented in this review propose reputation-based trust management for Grid computing environments that are primarily intended for QoS aware resource selection and allocation. There is a strong need to verify that none of the existing solutions model the *subjective* aspect of trust and therefore justify the need for a new reputation-based trust management system.

In practical terms, this implies allowing Grid client components (such as resource brokers, meta-schedulers and monitoring toolkits) to actively participate in the trust and reputation evaluation process utilising QoS negotiation means. This should be achieved by endeavouring an *exoteric* approach towards trust management, where Grid clients are prompted to stipulate reputation evaluation criteria to match their specific job requirements, the global trust context and attitude towards risk.

The reputation evaluation criteria should be comprised of a permutation of QoS aspects (e.g. availability, reliability, and etc) as well as relationship rules, such as weight factors between the QoS aspects and between direct and indirect sources. Moreover, the reputation evaluation criteria should be consolidated with *opportunistic* trust based decisions allowing Grid client applications to predefine a threshold of trustworthiness reflecting on their individual attitude to risk. An additional need is for the reputation evaluation criteria to support targeting at multiple levels such as resource, organisation and VO utilising trust aggregation techniques.

Based on the aforementioned requirements, Table 2.4 was created in order to summarise and compare the relevant properties of each of the reviewed solutions. As indicated by the table, none of the existing solutions model the *subjective* nature of trust as all listed solutions are based on esoteric reputation queries. Consequently, none of the existing solutions can be catered for different types of job requirements as all of them propose a central reputation service providing deterministic, predefined metrics for selecting a trusted resource from a list of possible alternatives.

Given the fact that reputation is composed of multiple aspects (e.g. availability, reliability, accuracy and etc) [92], Azzedin & Maheswaran [14, 15, 16], GridEigenTrust [28, 57, 30] and PathTrust [58] are considered limited as these solutions perceive reputation as single aspect. The GridPP [133, 134, 135] solution is considered

limited as well since its model aspects are finite and fixed (availability & reliability) with no consolidation support; thus confining the notion of multi-faceted reputation.

Other solutions, such as the ones proposed by Jia & Qu [59] address this limitation by treating reputation as multi-faceted and uncertain. Similarly to SeGO [60, 61], it employs fuzzy-logic inference for modelling the uncertainty aspect of trust. However, while SeGO proposes fuzzy-logic for reducing platform vulnerability and guiding the defence deployment across Grid sites, Jia & Qu accommodate a fuzzy partial ordering model for improving resource selection which is more relevant to the work described in this thesis. Nevertheless, they do not allow any control over the permutation of quality aspects (e.g. capability, honesty and reliability), the range and domain of the fuzzy partial-order sets used for evaluation nor ability for aggregating trust as their solution merely addresses reputation of resource providers.

The solution proposed by Aziz & Silaghi [53, 54] offers an exoteric feedback mechanism utilising a subjective utility function encompassing one or more issues of interest (e.g. expected QoS). In this approach, the reputation of an entity is built from the aggregation of the reputation of each of its issues of interest from the perspective of the user. While this mechanism is a step forward towards subjective trust, it lacks the ability for exoteric reputation querying as there is no control over the quality aspects. In addition, this solution associates between a user and a resource as users are required to supply a utility function with each job submission. This thesis discourages this approach as reputation management is perceived as a system level service (i.e. middleware infrastructure) which should be abstracted away from Grid users - not doing so will negatively impact the overall user experience.

Despite the research achievements in trust and reputation management in computational grids, there is still substantial work to be undertaken in this field. The adoption of multi-faceted trust as well as the utilisation of fuzzy-logic partial ordering are both welcome additions, as they allows addressing the complexity and uncertainty aspects of trust. In addition, the introduction of QoS and SLA negotiation between two parties (e.g. service consumer and service producer) further extends the frontier as it marks an important step in the direction towards subjective based trust management. However, as the literature review and the critical analysis concluded, none of the existing solutions support *exoteric* based reputation queries which are adapted to the criteria of the submitted job, the global trust context, risk aptitude and etc. In order to fill this gap and address the current shortcomings in the state of the art, a synergistic reputation-policy based trust model is required.

The proposed synergy functionality is required at two distinct levels: (i) usage of policy assertions to stipulate reputation evaluation requirements thus combining elements of reputation-based and policy-based trust models; and (ii) integration of a multi-faceted evaluation model with a fuzzy inference based decision model.

2.4.2 Rationale

In order to justify the need for a reputation-policy based trust model, an hypothetical scenario shall be used for demonstrating how it can be applied within the financial industry for supporting different types of intensive data operations.

A leading investment bank utilises an enterprise Grid as part of its underlying meta-computing IT infrastructure. This infrastructure is comprised of several computing and storage clusters distributed at multiple geographical regions and spanning multiple internal divisions of the bank. Each cluster is managed locally and characterised by varying levels of delivered service quality. For example, some clusters produce reliable computation results yet they tend to become non-available due to frequent CPU overloading. In contrast, other clusters are usually available however they tend to produce inaccurate results due to greedy configuration methods.

Several types of analytics jobs are executed on the enterprise Grid overnight. Two of them are the historical Volume Distribution Profile (VDP) generation and European stock options pricing. These two types of analytics jobs are very different in terms of their service quality requirements and therefore, different subsets of the infrastructure should be utilised in order for each job to run and complete successfully. The historical VDP generation involves segmenting each supported stock S trading day into several time intervals (bins) and computing the average percentage of daily volume traded during each specified bin (time interval). The output of this computation is a decimal number S_{bin_n} for each bin such that $S_{bin_n} \in [0,1]$. The accuracy of this number is highly crucial for the business divisions of the bank as it directly impacts investment decisions. However, if one or more bins are missing due to non-available resources, a previous day distribution profile can substitute the missing entries and result in a suboptimal yet acceptable solution.

In contrast, the European options pricing formula utilises a Monte-Carlo method for generating several thousand trajectories of price paths for an underlying stock S_T via pricing simulation, which results in a distribution of the stock price, S_T at time $t = T$ (i.e. the option exercise date). In order for this method to be valid, many trajectories have to be simulated so a large number of computing resources must be available. However, since option pricing is a stochastic process (i.e. involves generating random numbers) the accuracy of the computing resources is less relevant.

In this scenario, the Grid client application is likely be a resource broker. Given the current state of the art, the resource broker can be integrated with a reputation-based trust management system. This action upgrades the resource broker into a trust-aware resource broker, in which trust is integrated into resource management [15, 54]. The trust levels produced by the reputation-based trust management system can be considered by the resource broker for selecting the final list of resources. According to Chakrabarti [80], there are two common ways for making

trust-based decisions: *threshold-based* and *rank-based* functions.

In a threshold-based function, a predefined threshold t (where $0 \leq t \leq 1$) is compared against a list of evaluated resources R, such that if a resource trust level is tl_r is equal or greater than t (i.e. $tl_r \geq t$), it is appended to final list R' ($R' \subseteq R$).

$$
\begin{array}{|l}
\hline
Filter : R \rightarrow R' \\
\hline
\forall r \in R \bullet \\
Filter(r,t) = \begin{cases} 1 & \text{if } tl_r \geq t; \\ 0 & \text{if } tl_r < t; \end{cases}
\end{array}
$$

A rank-based function is useful when the calculated trust level of a single entity does not convey much information unless it is compared with the trust levels of other entities. In such function, the relative position of a resource $r(n)$ in a descending ordered list $Rank(R)$ (i.e. rank order) where $tl_{r(n)} \geq tl_{r(n+1)}$ is compared with a predefined percent p (where $0 \leq p \leq 100$) of the cardinality of R - i.e. $\#R$. This essentially translates to: select the first $p\%$ of resources out of the rank order of R.

$$
\begin{array}{|l}
\hline
Filter : R \rightarrow R' \\
\hline
\forall r \in Rank(R) \bullet \\
Filter(r,p) = \begin{cases} 1 & \text{if } r(n) \geq round(0.01p \cdot \#R); \\ 0 & \text{if } r(n) < round(0.01p \cdot \#R); \end{cases}
\end{array}
$$

Due to the current shortcomings in this field, none of the existing solutions would have been able to negotiate QoS based on the requirements of the submitted job. Effectively, none of the existing solutions would have allowed the resource broker to define reputation evaluation criteria targeting the relevant QoS aspects and rankings for the specific job. In addition, none of the existing solutions incorporates a decision model which allows mappings of trust values into trust levels. Given the limitations in the state of the art, both presented jobs (VDP generation and European option pricing) would have allegedly produced less than favourable results.

In an ideal scenario, the resource broker should have been able to provide reputation evaluation criteria thus adjusting the trust and reputation evaluation process. In this example, the resource broker could have been able to specify 0.9/0.1 reliability/availability weight ratio for the volume distribution profile generation job and 0.1/0.9 reliability/availability weight ratio for the European options pricing job and as a result, preemptively select relevant clusters and optimise both computations.

2.5 Conclusions

This chapter encompassed three objectives. The first objective was to discuss the background research activities including the emergence of trust management in various computing environments, the common methodologies for managing trust (i.e. policy-based and reputation-based) and review the key elements of reputation-based trust management systems including common trust evaluation methodologies.

The second objective was to discuss the related work regarding reputation-based trust management in Grid computing. This section initialised with a description regarding the evolution of trust management in computational grids. Consecutively, it continued with a literature review presenting the state of the art solutions currently available. This section finalised with a table summarising and comparing the properties of the existing solutions and highlighting their key differences.

The third objective is divided into three sections. The first section was to provide a critical analysis of the existing solutions. In the second section, a generalisation is made regarding the shared deficiencies of the existing solutions. Finally, in the third section a justification is made for a novel synergistic reputation-policy based trust model. In conclusion, the main outputs of this chapter are the following:

- Comprehensive background research identifying the key elements of reputation-based trust management systems and common trust evaluation methodologies.
- Generalisation of the deficiencies in the state of the art Grid reputation-based trust management systems and an identification of the main requirements in order to address these deficiencies.
- Solid justification for introducing a novel reputation-policy trust model based on the current deficiencies and the accompanied hypothetical scenario.

Chapter 3

Synergistic Reputation-Policy Based Trust Model

"Every great advance in science has issued from a new audacity of imagination."

John Dewey

3.1 Introduction

The purpose of this chapter is to introduce the *synergistic reputation-policy based trust model*, which is a key contribution of this research thesis (RC1). The main objective of this model is to address the deficiencies of existing models identified during the state of the art and critical analysis and consecutively offer a novel trust model in which its emphasis is on modelling the *subjective* aspect of trust.

Therefore, this chapter is divided into two sections. The first section discusses the envisioned requirements and characteristics that the reputation-policy trust model should possess describing its novelty points and advantages over existing trust models. The second section describes in great detail the anatomy of the model including internal artifacts, their key interactions required for producing trust level values and the main constraints (or boundaries) imposed on the model.

3.2 Toward Reputation-Policy Based Trust Management

3.2.1 Overview

As aforementioned, the existing Grid reputation-based trust management systems are typically evolved around community reputation services for computing resources trust values. The purpose of these reputation services is to enable an adequate

level of confidence when performing resource selection using single, deterministic reputation algorithms. The trust metrics calculation for these algorithms is *esoteric* thus limiting external involvement in the trust and reputation evaluation process. Advanced approaches in this field (such as the utility models [53, 54]) consider the multi-faceted characteristic of trust as well as SLA and QoS negotiation prior to service consumption [52]. However, none of the existing trust models offers a fine-grained methodology for exoterically evaluating trustworthiness of Grid computing resources, based on subjective interpretations of trust and adapted contextual factors such as the type of the submitted job, the global trust value, risk aptitude and etc.

This identified limitation has stimulated the motivation behind the synergistic reputation-policy based trust model for Grid resource selection. It is essentially a reputation-based trust model; however it is fortified with policy assertions utilised for stipulating reputation evaluation requirements. This introduction stands in contrast to policy-based trust management systems where policy assertions are utilised in the context of stipulating resource access control. As a result, this model forms a *synergy* between the elements of reputation-based and policy-based trust models.

The novelty of the synergistic reputation-policy based trust model lies in its inherent ability to encourage an active participation in the trust and reputation evaluation processes. This is achieved by enabling Grid clients (such as resource brokers, meta-schedulers and monitoring systems) to extend their existing standard reputation queries with a set of reputation-policy assertions constituting as trust decision strategies. As a central aspect of the model, each trust decision strategy is comprised of an evaluation model represented by a permutation of opinion elements (QoS aspects) as well as a decision model comprising of decision rules describing mappings between trust values and correspondent trust levels. Together they form a complete trust metrics blueprint for the reputation algorithm. Figure 3.1 illustrates the concept of a *trust decision strategy.*

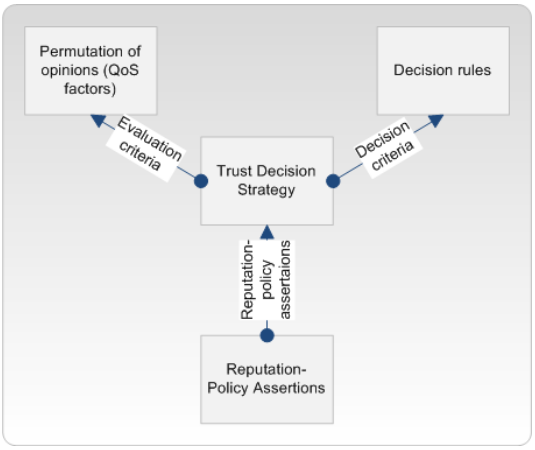


Figure 3.1: Concept of a Trust Decision Strategy

In practical terms, this implies that Grid clients should be supplied with means for evaluating the *trustworthiness* of Grid computing resources by articulating subjective trust based decisions reflecting their intrinsic view on trust and the type of job they wish to submit. For example, two Grid client programs may have different opinions and decision actions regarding the trustworthiness of the same Grid resource, given their job requirements, the global trust context and evaluation criteria. This behaviour is based on the theorem described in the problem statement of this thesis proclaiming that a reputation-policy based trust model would allow an elevated degree of fine-grained resource selection and consecutively optimise the overall quality of computations by reducing the rate of job execution failures.

3.2.2 Reputation-Policy Concepts

Characteristics

The philosophy behind the reputation-policy trust model is that reputation is perceived as subjective matter as well as context dependent. This stands in accordance with the theoretical work made by Gambetta [13] who 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 as well as Jøsang et al. [22] who developed the *subjective logic* for modelling and analysing situations involving uncertainty and incomplete knowledge. The reputation-policy subjective behaviour is reflected in both the evaluation of entities as well as in making trust based decisions.

The evaluation of entities in the reputation-policy trust model is based on elements from discrete models proposed by Azzedin and Maheswaran [14, 15, 16] - i.e. calculating trust values by combining direct and indirect aggregations of reputation feedback ratings. However, the reputation-policy trust model extends the standard discrete models to reflect the multi-faceted aspect of trust, so separate trust value calculations are performed for each segregated aspect (i.e. availability, reliability and etc.) and then combined thus in order to produce the overall trust value. This stands in accordance with both Jia & Qu [59] as well as Aziz & Silaghi [54] who implemented the multi-faceted aspect of trust as part of their solutions. The main difference is that they incorporate multi-faceted trust as part of the SLA metrics (i.e. provider-promised factors) whereas the reputation-policy trust model incorporates it as part of the reputation evaluation criteria (i.e. consumer-expected factors).

Incorporating a multi-faceted discrete model for the evaluation of entities serves the advantage of attaining a fine degree of *exoteric* control over which context to evaluate against, which quality aspects to address and prioritise, which trust sources to blend (experience and/or reputation), which Trust Decay Function (TDF) to include and etc. yet maintain the coherence and comprehensibility of discrete models.

Trust based decisions in the reputation-policy trust model are achieved by utilising a *fuzzy inference system* [82]. While Hwang & Song [60] and Jia & Qu [59] propose fuzzy-logic for reducing platform vulnerability and improving resource selection respectively, the reputation-policy trust model endorses fuzzy-logic for *opportunistically* making trust based decisions based on risk aptitude and the global context. The utilisation of fuzzy-logic for the decision model serves the advantage of dealing with uncertainty and complexity associated with the decision making process.

Trust levels are normalised in the reputation-policy based trust model as a continuous value between 0 and 1 and therefore aligned with most related work. This also serves the advantage of providing some form of probabilistic measurement for trust and reputation. The combination of a discrete Evaluation Model and a fuzzy-logic based Decision Model formulates the synergistic reputation-policy paradigm over a consolidated *fuzzy-discrete* trust model which provides both trust evaluation as well as the management of trust based decisions. This is in accordance with Jøsang et al. [24] who perceive the notions of trust and risk as tightly coupled and therefore sees the need to formalise the relationship between the two aspects.

Assumptions

1. A collection of third-party recommendation agents exists in which each recommendation agent periodically submits reputation feedback ratings. This could possibly take the form of a monitoring system measuring and reporting the delivered QoS for each evaluation requested quality aspect.

2. All entities (i.e. users, resources and services) have been previously mutually authenticated and authorised in order to interact with each other.

3. All interacting trusting agents trust the intentions and the recommendations (i.e. non-biased ratings) produced by the reputation-policy trust model.

4. The creation, selection and management of relevant trust decision strategies is the exclusive responsibility of the trusting agents. The trusting agents are required to decide on the job reputation requirements and weigh it against the intended execution capacity, the depth of feedback ratings the global trust context and risk aptitude prior to submitting a trust decision strategy.

5. The trust level values provided by the reputation-policy trust model are merely recommendations. Therefore, the final resource selection should be subjected to additional constraints (e.g. software, hardware, capacity and etc.) and possibly filtered by either a threshold-based or a rank-based function.

Constraints

1. The reputation-policy trust model does not support reputation feedback queries as it relies on recommendation agents for submitting feedback ratings.

2. The reputation-policy trust model is intended for rating trusted agents (e.g. resources and services) and not users.

3. The reputation-policy trust model does not consider the credibility of the recommender RTF) as it assumes all recommendations are biased free.

4. No default trust level value is provided in case there are no feedback ratings.

5. The quality aspects listed in the trust decision strategies must not contain aspects which are not defined in the Customised Matrices Pool (CMP).

Considerations

The reputation-policy trust model considers the several characteristics, which are illustrated in Figure 3.2. The envisioned characteristics include the following:

– **Aggregation of reputation criteria:** The most predominant consideration of this model revolves around the definition of trust and how to model trust using policy assertions. As reputation criteria varies given different job requirements and perception of trust, the model is required to adapt to various scenarios and allow multiple interpretations by different trusting agents. Given the multi-faceted aspect of trust, the solution proposed in this research breaks down trust into a permutation of opinion elements (i.e. quality aspects). Each opinion is considered as atomic building block of trust and a collection of opinions constitutes as single view on trust. Grid clients can assemble different views using different sets of opinions and setting weights between them. The aggregation of opinions defines an evaluation strategy and a collection of evaluation models can be aggregated inside a (TDS repository).

– **Level of aptitude:** The reputation-policy trust model considers the degree of aptitude trusting agents acquire prior to submitting trust decision strategies. Strategies can be assembled either statically (via a central repository) or dynamically. In both cases, the decision on which evaluation strategy to submit is dependent on the information available in the reputation-policy data store. For example, if a strategy contains an evaluation model which dictates high importance on availability as an opinion factor, trusting agents which intend to use this strategy for evaluating resources, should have means of acquiring

if availability feedback information is available and the length of this information before deciding on submitting the strategy. This information should be made available through data mining techniques and allow it to be queried before engaging with reputation evaluation. The reputation-policy trust model considers this requirement for submitting optimised strategies, however, it assumes that when trusting agents submit strategies, they have previously selected a relevant strategy based on the available information.

– **Level of abstraction:** The reputation-policy based trust model does not require user involvement in manually selecting and submitting trust decision strategies. Instead, the bulk of the work is shifted to the trusting agent which acts on behalf of its users by - (i) contacting the TDS repository, (ii) obtaining the relevant TDS for the given type of job, (iii) submitting a reputation query and (iv) obtaining the results. This serves the advantage of directing an implementation of this model as a system level service which abstracts the details away and therefore enhances the overall user experience.

– **Extensible framework:** The reputation-policy trust model considers different VOs which may have different requirements based on the context of the computation. For example, a mission critical financial computation VO may have different reputation evaluation requirements from an e-science research VO which, is more likely to be tolerant to execution failures. As a result, the reputation-policy trust model allows extending its evaluation framework through the following means: (1) Allowing VO administrators to define the opinion factors available for evaluation, (2) Definition of new rules for reputation evaluation; and (3) Definition of new rules for trust decision making.

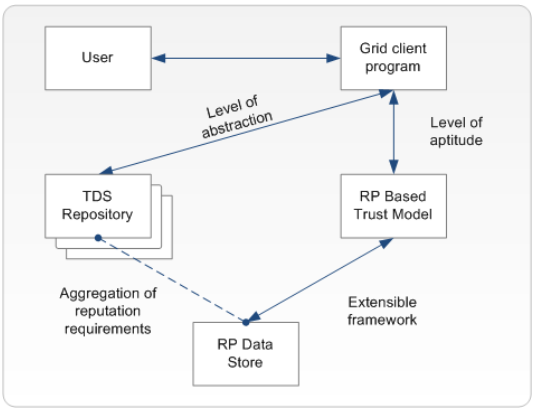


Figure 3.2: Reputation-Policy trust model considerations

3.3 Synergistic Reputation-Policy Based Trust Model

3.3.1 Overview

The key contribution of this research is promoted by the synergistic reputation-policy trust model (RC1). In contrast to existing esoteric reputation-based trust models, the reputation-policy trust model encourages external involvement in the trust and reputation evaluation process. It promotes the use of well-defined policy grammar to stipulate reputation evaluation requirements for each submitted job and therefore it creates a synergy between the principles of policy and reputation thus resulting in finer grained control over making trust based decisions.

Actors

The reputation-policy trust model acknowledges three predominant actors: trusting agent, trusted agent and a third-party recommendation agent:

- The *trusting agent* is defined as an entity who has faith or belief in another entity in a given context and at a given time slot. In the context of this dissertation, the trusting agent is a *resource consumer*, implemented as a software component that acts on behalf of a human user or another software component.
- The *trusted agent* is defined as an entity in whom faith or belief has been placed by another entity in a given context and at a given time slot. In the context of this thesis, the trusted agent is a *resource provider*, implemented as a software component that provides access to a service or computing resource.
- The *third-party recommendation agent* is defined as an entity which provides a reputation feedback regarding a trusted agent in a given context and at a given time slot. The feedback is comprised of an opinion and a feedback value.

Requirements

The reputation-policy trust model is required to provide functionality for evaluating trusted agents trustworthiness and for submitting reputation feedbacks. Figure 3.3 illustrates a high level use case diagram of the reputation-policy trust model. It depicts the actors and the two main use cases served by the model: *Evaluate Trusted Agent Trustworthiness* and *Submit Rating Feedback*. Both use cases include the *Obtain Reputation-Policy Evaluation Requirements* use case as it is required in both the evaluation of trusted agents as well as for submitting recommendations.

The trusting agent is required to contact the model in order to evaluate the trustworthiness of one or more trusted agents, an organisation or an entire VO. This includes obtaining the reputation-policy evaluation requirements and contacting one

or more third-party recommendation agents in order to collect their feedbacks regarding their historical experiences with the evaluated trusted agents.

Once a trusting agent is bound to a trusted agent, the trusting agent notifies the third-party recommendation agent which monitors the trusted agent performance (outside the scope of the model). Once the interaction between the third-party recommendation agent and the trusted agent is complete, the third-party recommendation agent submits a reputation feedback which measures the quality of the interaction. In order to rate interactions, the third-party recommendation agent should compare the level of service provided by the trusted agent to the quality of service guaranteed in the service level agreement (SLA) using a utility function.

The model does not concentrate on this aspect as it assumes the feedback values are generated externally. However, it does allow submitting reputation feedbacks regarding different quality factors. The third-party recommendation agent uses the reputation-policy evaluation requirements previously defined for selecting the trusted agent in order to determine which quality factors to submit reputation feedbacks for.

For example, if the reputation-policy requirements contained *availability* and *reliability* as quality factors, then the recommendation agent would have been required to submit reputation feedbacks for each of these factors. The feedbacks themselves should be first compared with the counterpart SLA metrics provided by the trusted agent and then weigh against the historical reputation feedbacks, such that the greater depth of historical interactions with the trusted agent the less impact on the final feedback value. This approach is advantageous to existing feedback formulae as it considers both the multi-faceted characteristic of trust as well as blending historical feedback ratings to the standard SLA projected vs delivered delta calculation.

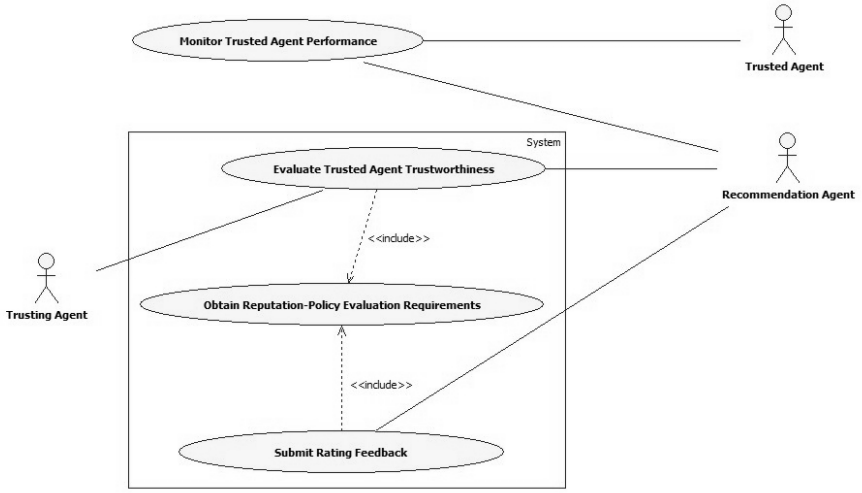


Figure 3.3: Reputation-policy trust model use case diagram

3.3.2 Model Structure

Overview

The reputation-policy trust model is based on a distributed data model where trust data is divided between the trusting agent and the recommendation agents. The trusting agent stipulates its reputation-policy requirements in a form of a trust decision strategy document while the historical reputation feedbacks provided by the recommendation agents are aggregated into opinion matrices data structures.

When calculating reputation, the reputation algorithm initiates a correlation process between the opinions defined in the trust decision strategy and the opinion matrices. This process involves a reconciliation of each opinion element (i.e. QoS aspect) defined in the trust decision strategy with it's historical information counterpart in order to compute the trust values. Each trust value $tv \in [0,1]$ is defined as the average of the aggregated historical reputation feedbacks made by recommendation agents weighted by the opinions and rules defined in the trust decision strategy. This value represents the general confidence over the competency of a particular trusted agent adapted to the trusting agents *subjective* perception of trust.

Once the correlation process is complete, the trust values are routed through a Decision Rule Engine (DRE) in order to quantify the corresponding trust level values. The purpose of this step is to provide a decision metric that assists in determining the amount of trust that the trusting agent should have in the trusted agent. This metric takes into account the trust values produced via the Correlation Process (CP) as well as the mapping and decision rules defined in the trust decision strategy. It represents the overall level of trust the trusting agent should have in a trusted agent adapted to the trusting agent opportunistic characteristic of trust. In addition, it places semantics or linguistic definitions to provide an approximate meaning expressed in natural language to each level of trustworthiness.

Finally, an analytics report is generated containing trust data analysis for each evaluated trusted agent. The analysis contains a report entry for each trusted agent including the trust value, the trust level and an analysis of each linguistic definition and its degree of membership. The trusting agent processes the report and examines the obtained trust level to determine the degree of which it should trust a particular trusted agent. It is important to note that the produced trust level value is simply a recommendation over the trustworthiness of the trusted agent. Practically, the trust level should be combined with additional factors which are outside the scope of the model (e.g. software packages, hardware, number of CPUs, memory and etc.) prior to making a final decision whether to engage with a trusted agent or not.

Process Analysis

Figure 3.4 illustrates a high level interaction between the trusting agent and the reputation algorithm when evaluating trusted agent trustworthiness. During **steps 1-3**, the trusting agent obtains the TDS from the TDS repository. It is important to note that the model does not impose specific storage facilities so strategies can be persisted and retrieved by different storage mediums such as databases, local or remote file systems. During **steps 4-5**, the trusting agent constructs a reputation-policy query (RPQ) which contains the TDS as well as a list of trusted agents to evaluate as well as context and time information. The query is then submitted to the Reputation Algorithm (RA) for processing. During **steps 6-8**, the reputation algorithm contacts the Correlation Process (CP) which in turn contacts the opinion matrices (not illustrated) in order to generate a trust value for each evaluated trusted agent. The opinion matrices interface an underlying data store which contains the historical reputation feedbacks supplied by recommendation agents. The trust value is generated by contacting each opinion matrix individually to return the trust value for a single aspect (i.e. availability, reliability and etc) and then accumulating the results based on the evaluation criteria in the (TDS). During **steps 9-10**, each trust value is routed to the Decision Rule Engine (DRE) in order to generate a corresponding trust level value. The DRE dictates the usage of fuzzy logic reasoning for mapping between trust values and trust levels. During **steps 11-12**, The Reputation Algorithm (RA) generates a Reputation-Policy Report (RPR) and returns it back to the trusting agent which makes a trust based decision based on the analysis documented in the report. The following subsections describe the artifacts.

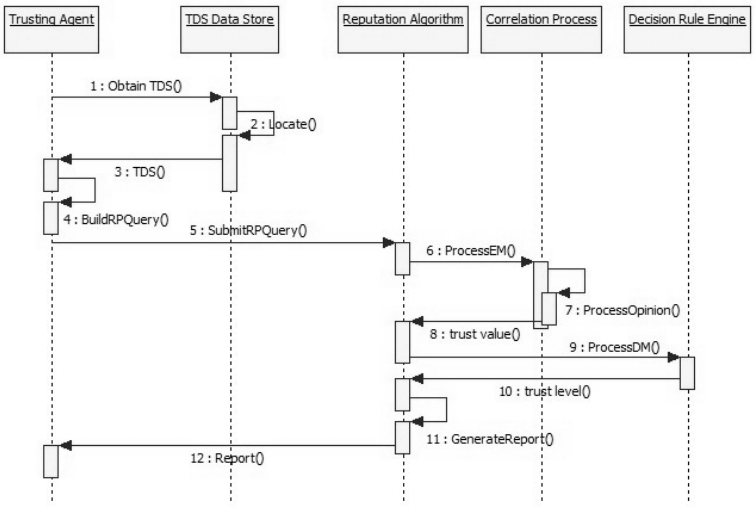


Figure 3.4: Reputation-policy trust model sequence diagram

Internal Artifacts

Figure 3.5 illustrates the internal artifacts of the reputation-policy trust model. It contains six elements: (i) Reputation-Policy Query (RPQ), (ii) Reputation Algorithm (RA), (iii) Correlation Process (CP), (iv) Customised Matrices Pool (CMP), (v) Decision Rule Engine (DRE) and (vi) Reputation-Policy Report (RPR). These are described in the following subsections using the Z notation [116].

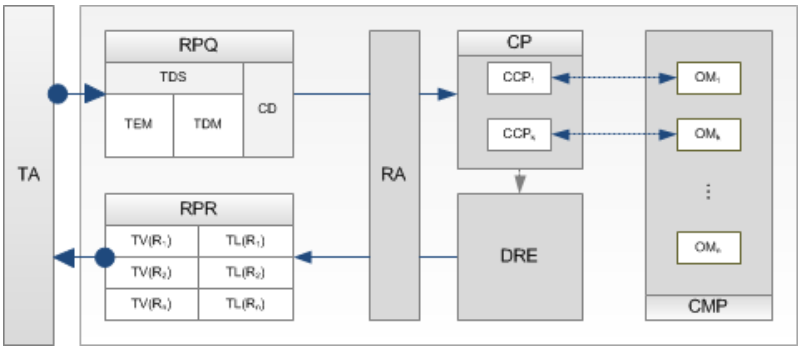


Figure 3.5: Reputation-policy trust model internal artifacts

Definition 1: Reputation-Policy Query

The *Reputation-Policy Query* (RPQ) is an extended reputation query which contains a trust decision strategy as well as context data. It is an inquiry made by a trusting agent for quantifying the trust level of one or more trusted agents. The inquiry is made for a specific context (i.e. store data) and a time slot.

ReputationPolicyQuery

$CD : ContextData$

$TDS : TrustDecisionStrategy$

Figure 3.6 illustrates the structure of the Reputation-Policy Query (RPQ). The query - *RPQ* is composed context data - *CD* and trust decision strategy - *TDS*.

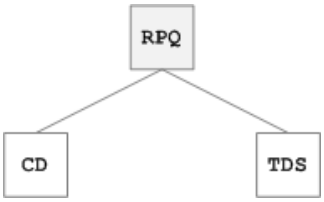


Figure 3.6: Reputation-Policy Query building blocks (RPQ = Reputation-Policy Query, CD = Context Data, TDS = Trust Decision Strategy)

Definition 1.1: Context Data

The *Context Data* (CD) artifact contains the following parameters:

- **Context ID** - the identifier of the context for which the reputation query is made for (e.g. store data, computation service and etc).
- **Trusting agent ID** - The identifier of the trusting agent which is used for validating the agent making the request as well as for differentiating between the trusting agent and peer agents when processing the evaluation model.
- **Trusted agent(s) ID** - The identifiers of the trusted agents. This field can also be extended to include an organisation (collection of trusted agents) and even en entire VO (collection of organisations).
- **Cut-off datetime** - the start date of which to gather the feedback data. A null value assumes to use the earliest date a feedback was ever submitted
- **Trust decay function** - the rate of trust decay. This results in assigning a weight to each submitted feedback given greater importance to feedbacks submitted closer to the time the query was made. the trust decay function is completely exoteric so that the trusting agent can supply virtual any type of function ($f(x) = 1/x, f(x) = 1/x^2$ and $f(x) = exp(x)$ and etc).

$[ID, DATETIME, TRUSTDECAYFUNCTION]$

ContextData

$contextID : ID$

$trustingAgentID : ID$

$trustedAgentIDs : \mathbb{P}\, ID$

$cutoffDateTime : DATETIME$

$trustDecayFunction : TRUSTDECAYFUNCTION$

Definition 1.2: Trust Decision Strategy

The TDS artifact is a structured document containing an aggregation of reputation evaluation requirements as well as trust decision rules. This document is submitted by the trusting agent to the Reputation Algorithm (RA). These requirements stipulate the desired QoS aspects to be evaluated and their weight factors as well mappings between trust values and correspondent trust levels. It is comprised of: (i) *Evaluation Model* and (ii) *Decision Model*.

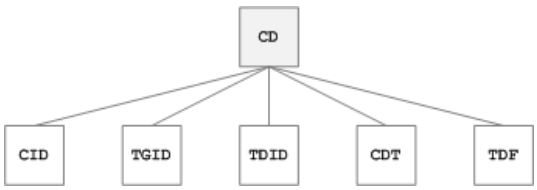


Figure 3.7: Context Data building blocks (CD = Context Data, CID= Context ID, TGID = Trusting Agent ID, TDID = Trusted Agents IDs, CDT = Cutoff DateTime, TDF = Trust Decay Function

TrustDecisionStrategy

$EM : EvaluationModel$

$DM : DecisionModel$

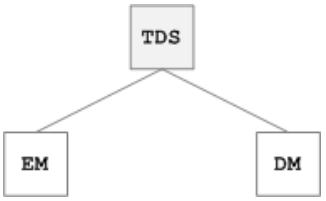


Figure 3.8: Trust Decision Strategy building blocks (TDS = Trust Decision Strategy, EM = Evaluation Model, DM = Decision Model)

Definition 1.2.1: Evaluation Model

The *Evaluation Model* (EM) is a discrete trust model based on the theoretical work made by Azzedin and Maheswaran [14, 15, 16]. However, it extends their model to support multi-faceted trust. The purpose of the evaluation model is to support the *subjective* aspect of trust. It is used by the correlation process to contact the opinion matrices and generate a trust value for each evaluated trusted agent.

EvaluationModel

$opinions : \mathbb{P}\, Opinion$

$(\textbf{let}\ o == Opinion \bullet o \in opinions;\ )$

$\sum o.weight \in opinions = 1$

Figure 3.9 illustrates the structure of an evaluation model. The evaluation model *EM* is comprised of a finite set of opinions $(O_1, \ldots, O_n)$. There are two logical constraints regarding the evaluation model: (i) Each opinion o in $\mathbb{P}\, Opinion$ must be unique such as: $\forall i,j \in \{1,\ldots,n\}, (i \neq j) : o_i \neq o_j$; and (ii) the summation of all

opinion weights must be equal to 1 - i.e. $\sum o.weight \in opinions = 1$.

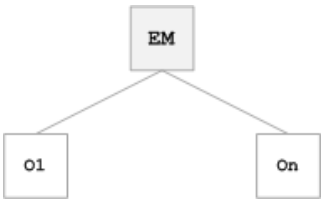


Figure 3.9: Evaluation Model building blocks (EM = Evaluation Model, O = Opinion)

Definition 1.2.1.1: Opinion

An *opinion* is an atomic building block of trust which reflects a single QoS aspect (e.g. availability, reliability, data accuracy and etc). The VO administrator defines a set of opinions applicable for the VO during the identification phase. Each defined opinion in the set corresponds to an historical feedback ratings counterpart - Opinion Matrix (OM) which is located inside a Customised Matrices Pool (CMP). Therefore, the opinion elements stipulated in the EM must be a subset of the opinions defined by the VO administrator. Let o define an opinion, O_{EM} defines a set of opinions defined in the EM and let O_{CMP} denote a set of opinions defined by the CMP. The opinion inclusion constraint is defined as: $O_{EM} \subseteq O_{CMP}$. An opinion O is defined as:

$$OPINIONTYPE ::= availability \mid reliability \mid data\_accuracy \mid etc.$$

$$fuzzy\_value : 0 \,..\, 1$$

Opinion

$$type : OPINIONTYPE$$
$$weight : OPINIONTYPE \rightarrow fuzzy\_value$$
$$sources : \mathbb{P}\, Source$$

$$(\textbf{let } s == Source \bullet s \in sources;)$$
$$\sum s.weight \in sources = 1$$

Where $type \in \mathbb{N}_1$ defines the type of the opinion (i.e. availability, reliability and etc.), $weight \in [0,1]$ defines the weight of the opinion (i.e. importance) and *sources* defines the trust sources. Figure 3.10 illustrates the structure of an opinion. The opinion O is comprised of its type T_o, its weight W_o and a source set S.

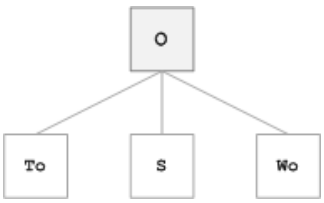


Figure 3.10: Opinion building blocks (O = Opinion, To = Type of opinion, S = Source set, Wo = Weight of opinion)

Definition 1.2.1.1.1: Source

A *trust source* is a reference for information such as reputation or experience. The source defines the gathering location for an opinion. An experience source means the self experience the trusting agent had achieved with the trusted agent whereas reputation means (third-party) peer agents experiences with the trusted agent. Sources for an opinion can have a weight factor, indicating the importance of one source type over another source type.

$$SOURCETYPE ::= experience \mid reputation$$

Source

$$type : \mathbb{P}\, SOURCETYPE$$
$$weight : SOURCETYPE \rightarrow fuzzy\_value$$

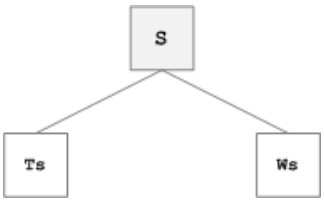


Figure 3.11: Source building blocks (S = Source set, Ts = Type of source, Ws = Weight of source)

In general, weight factors form part of a larger concept known as relationship rules. *Rules* are general constraints which can be attached to elements (e.g. decisions, opinions and sources) or group of elements. Rules are modelled using fuzzy logic, indicating a degree of influence of one rule over another. The *weight* for a source is modelled using a fuzzy value indicating the degree of importance where 1 represents complete importance, and 0 for irrelevance. Similarly to sources, opinions can have weight factors associated with them, indicating an importance of an opinion over

another. A set of n opinions constitutes as an evaluation model forming the ontology aspect of the reputation policy. The output of the evaluation model is a value referred to as *trust value* and it serves as an input to the decision model.

In summary, the EM consists of a set of opinions where each opinion is associated with an opinion type (QoS factor) and opinion weight (importance). In addition each opinion contains a set of sources, where each source is associated with a source type (gathering location) and source weight (importance). Using the evaluation model, two trusting agents can potentially receive different trust values for the same trusted agents due to the opinions, sources and weights defined in each model.

Definition 1.2.2: Decision Model

The *Decision Model* (DM) contains an aggregation of trust decision rules. It is based on a Fuzzy Inference System (FIS) for mapping the trust values produced by the correlation process into trust levels. The purpose of the decision model is to support the *opportunistic* aspect of trust by taking an advantage of various circumstances (e.g. risk aptitude, global context and etc) for determining the trustworthiness of a trusted agent. Figure 3.12 illustrates the structure of the decision model. It contains three aspects: (i) Fuzzifier Set, (ii) Defuzzifier and (iii) Decision Rule Set:

$$
\begin{array}{|l}
\hline
\textit{DecisionModel} \\
\textit{FZS} : \mathbb{P}\, \textit{Fuzzifier} \\
\textit{DZ} : \textit{Defuzzifier} \\
\textit{DRS} : \mathbb{P}\, \textit{DecisionRule} \\
\hline
\end{array}
$$

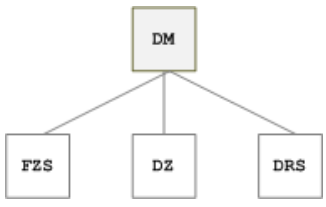


Figure 3.12: Decision Model building blocks (FZS = Fuzzifier Set, DZ = Defuzzifier, DRS = Decision Rule Set)

Definition 1.2.2.1: Fuzzifier

The *fuzzification* allows the definition of one or more membership functions for a given input variable (e.g. trust value or context value). It is comprised of a fuzzifier name - i.e. the name of the input variable and a set of terms (TS). Each term specifies a membership function with a distinct name (e.g. 'poor', 'good', 'excellent') and a

series of axis points defining the length and scale of the membership function.

$[TERM]$
$FUZZIFIERNAME ::= trust\_value \mid context\_value$

Fuzzifier
$FN : FUZZIFIERNAME$
$TS : \mathbb{P}\, TERM$

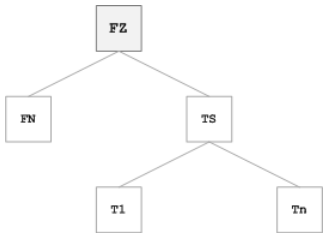


Figure 3.13: Fuzzifier building blocks (FZ = Fuzzifier, FN = FuzzifierName, TS = Term Set)

Definition 1.2.2.2: Defuzzifier

The *defuzzification* aspect is the process of producing a quantifiable trust level values by using the defined decision rules and applying a defuzzification method such as CoG (Center of Gravity) or MoM (Middle of Maximum).

$ACCUMULATIONMETHOD ::= max \mid bsum \mid nsum$
$DEFUZZIFICATIONMETHOD ::= cog \mid cogs \mid coa \mid lm \mid rm$

Defuzzifier
$AM : ACCUMULATIONMETHOD$
$TS : \mathbb{P}\, TERM$
$DM : DEFUZZIFICATIONMETHOD$

Definition 1.2.2.3: DecisionRule

A *DecisionRule* (DR) is defined as an if-then statement. All decision rules are evaluated in parallel, and the order of the rules is unimportant. The rules themselves are useful because they refer to variables and the adjectives that describe those

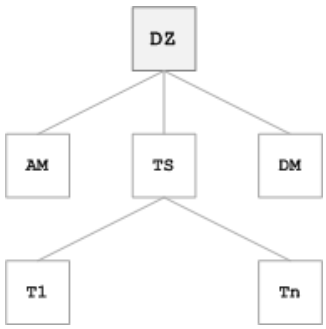


Figure 3.14: Defuzzifier building blocks (DZ = Defuzzifier, AM = Accumulation Method, TS = Term Set, DM = Defuzzification Method)

variables. For example the following expression constitutes as a rule for mapping between an "'EXCELLENT"' term for trust value and a "'HIGH"' term for trust level: "'IF TrustValue is EXCELLENT THEN TrustLevel IS HIGH"'.

DecisionRule

$ruleID : ID$

$expression : ID \rightarrow TEXT$

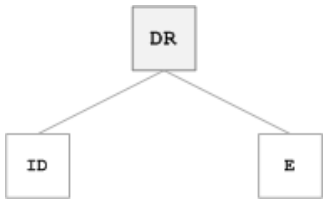


Figure 3.15: DecisionRule building blocks (ID = RuleID, E = Expression)

Figure 3.16 illustrates the overall structure of a Reputation-Policy Query (RPQ) comprising of Context Data (CD) and a Trust Decision Strategy (TDS). The TDS is comprised of an Evaluation Model (EM) as well as a Decision Model (DM) thus forming complete trust metrics for the Reputation Algorithm (RA).

Definition 2: Reputation Algorithm

The *Reputation Algorithm* (RA) artifact acts as a façade to the Correlation Process (CP). It accepts as an input a Reputation-Policy Query (RPQ) and returns a Reputation-Policy Report (RPR) as a result of the computation. The purpose of the reputation algorithm artifact is to facilitate the interaction between the trusting agent with both the CP and the Decision Rule Engine (DRE). The RA can be used to facilitate the processing of an RPQ by: (i) validating and deserialising the reputation query data, (ii) generating and serialising the RPR.

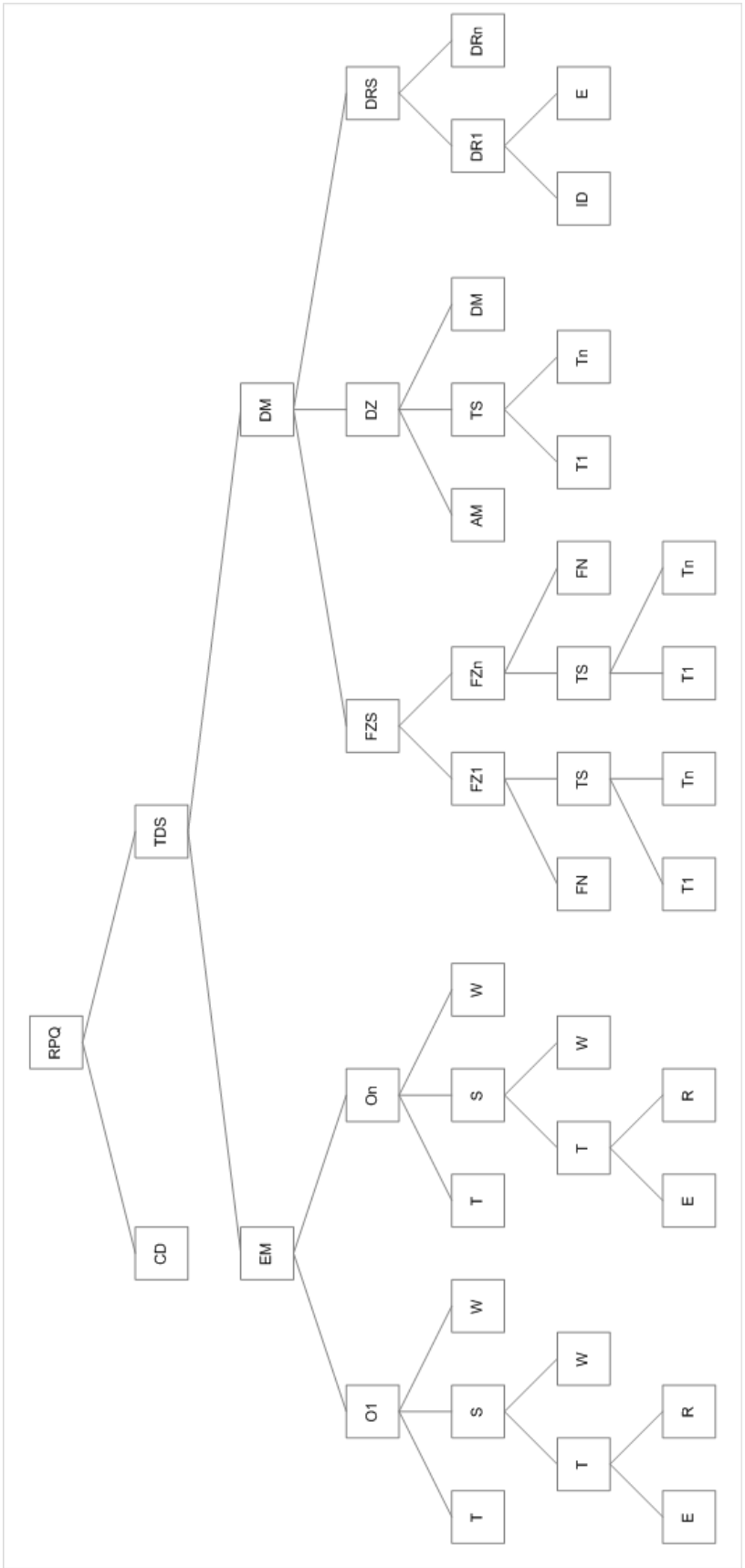


Figure 3.16: Overall Structure of a Reputation-Policy Query

ReputationAlgorithm

$process : ReputationPolicyQuery \rightarrow ReputationPolicyReport$

Definition 3: Correlation Process

The *Correlation Process* (CP) involves matching each opinion defined in TDS with its historical references in the opinion matrices and calculating the trust value for that opinion. The correlation process is designed as a multi-threaded algorithm. It is supplied with three arguments: (i) set of opinion types, (ii) set of resources (iii) and context data (CD) information extracted from the reputation query. Processing instructions dictate context factors (e.g. Cutoff Time (COT), Trust Decay Function (TDF), etc.) which influence the values contained by an Opinion Matrix (OM). The generation of trust value for each evaluated resource is comprised of three steps:

- **OM initialisation** - For each opinion type defined in the EM, the CP forks a *Child Correlation Process* (CCP). Each TDS opinion type is routed via CMP in order to return a correspondent OM. For example, if the EM defined availability and reliability as opinion types, the CP will allocate two correspondent opinion matrices. The values in each opinion matrix are dictated by the opinion type routed to the CMP, the volume of historical reputation feedbacks and the contextual data used for controlling the underlying data store.

- **OSM population** - The CCP operates on the opinion matrix and computes the overall trust value for the particular opinion type (e.g. trusted agent: 1 opinion: availability value: 0.83). This is achieved by using the opinion sources (reputation and experience) and accumulating the opinions from the different third-party recommendation agents and applying the source weight rules on each agent (if the third-party recommendation agent is the trusting agent, then the experience rule is applied). Each opinion based trust value is stored inside an Opinion Summary Matrix (OSM). The OSM consists of a set of evaluated trusted agents as rows and opinion based trust values as columns.

- **RVT generation** - This step commences once all child correlation processes have returned to the parent process. The CP computes the overall trust value by accumulating the trust value for each trusted agent opinion type and multiplying is with the opinion weight rule using a weighted mean. Each resource and value pair are inserted into a hash table known as Resource Value Table (RVT). Once the RVT generation is complete, the CP returns it back to the reputation algorithm for trust decision processing and report generation.

CorrelationProcess

$process : \mathbb{P}\, Opinion \rightarrow ResourceValueTable$

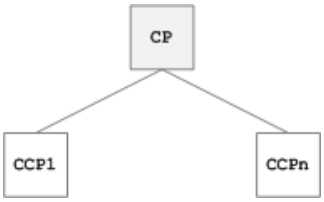


Figure 3.17: Correlation Process building blocks (CP = Correlation Process, CCP = Child Correlation Process)

Definition 4: Customised Matrices Pool

The *Customised Matrices Pool* (CMP) artifact stores a collection of *opinion matrices* (OM) thus facilitating access to a particular opinion matrix by a child correlation process. Each opinion matrix is mapped to a single opinion (e.g. the availability opinion matrix is mapped to the availability opinion). It is important to note that the model does not dictate an underlying data store for the opinion matrices so different storage mediums can be used (e.g. database tables, file systems and etc).

CustomisedMatricesPool

$OMS : \mathbb{P}\, OpinionMatrix$

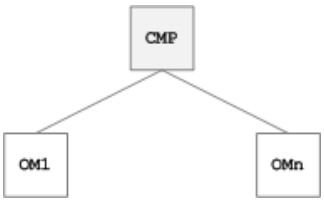


Figure 3.18: Customised Matrices Pool building blocks (CMP = Customised Matrices Pool, OM = Opinion Matrix)

Definition 4.1: Opinion Matrix

An *Opinion Matrix* (OM) is a tabular data structures which stores the historical reputation feedbacks submitted by third-party recommendation agents on transaction completion. For each opinion defined in the CMP universe there is one and only one correspondent OM, storing reputation feedback data regarding that opinion. This characteristic is of particular importance during the CP, where each

opinion defined by inside the EM is matched with its OMs counterpart.

After an execution completes, a trusting agent is required to rate the quality of the transaction using an evaluation feedback mechanism. This mechanism gathers a score value for each opinion originally defined by the trusting agent using the trust decision strategy. A replica of the evaluation feedback is stored by the Customised Matrices Pool (CMP) in order to be utilised by the opinion matrices.

The general matrix model $M(O)$ for an opinion contains columns $\{C_1,\ldots,C_j\}$ representing a set of trusted agents and rows $\{R_1,\ldots,R_i\}$ representing a set of trusting agents. It is important to note that the set of the trusted agents $\{C_1,\ldots,C_j\}$ is evaluated against an identical context (.e.g store data) and opinion (e.g. availability). The fuzzy value of each cell inside an opinion matrix represents a computed trust value for an opinion over n executions between a trusting agent and a trusted agent. It is important to note that the value can also be null. Typically, the value will be computed using standard mean. However, the computation of the value can be controlled by the trusting agent using cut-off time and a trust decay function.

$$M(O) = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \ldots & v_{1,j} \\ v_{2,1} & v_{2,2} & v_{2,3} & \ldots & v_{2,j} \\ v_{3,1} & v_{3,2} & v_{3,3} & \ldots & v_{3,j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{i,1} & v_{i,2} & v_{i,3} & \ldots & v_{i,j} \end{bmatrix}$$

The fuzzy value $v_{[i,j]}$ represents a computed trust value for an opinion over n executions between trusting agent α and trusted agent β. It is important to note that $v_{[i,j]}$ can be null. Normally, $v_{[i,j]}$ will be computed using standard mean:

$$v_{[i,j]} = \frac{1}{n}\sum_{a=1}^{n} V_a(\alpha,\beta,O) \tag{3.1}$$

Definition 5: Decision Rule Engine

The *Decision Rule Engine* (DRE) represents a trustworthiness scale. It is based on a fuzzy inference system for mapping an arbitrary trust value (and optionally - a context value) into a trust level based on the supplied decision model policies.

DecisionRuleEngine

$tv : fuzzy\_value$

$cv : fuzzy\_value$

$decisionModel : DecisionModel$

$process : (tv, cv) \rightarrow tl$

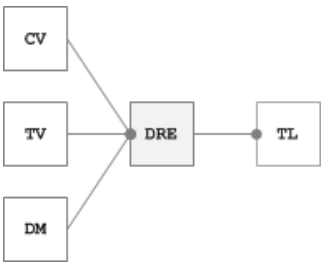


Figure 3.19: Decision Rule Engine building blocks (DRE = Decision Rule Engine, TV = Trust Value, CV = Context Value, DM = Decision Model, TL = Trust Level)

Definition 6: Reputation-Policy Report

The *Reputation-Policy Report* (RPR) artifact is the final generated output of the reputation-policy trust model. The report is comprised of a collection of report items $(RI_i \dots RI_n)$ where each report item $RI_k, (i \leq k \leq n)$, contains a trust value and level pair (tv_k, tl_k) corresponding to each evaluated trusted agent (Figure 3.20).

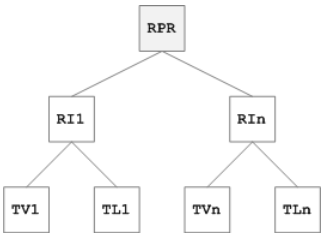


Figure 3.20: Reputation-Policy Report building blocks (RPR = Reputation-Policy Report, RI = Report Item, TV = Trust Value, TL = Trust Level)

ReputationPolicyReport

$\Xi CustomisedMetricsPool$

$items! : \mathbb{P}\, ReportItem$

Definition 6.1: Report Item

As aforementioned, a *Report Item* (RI) entity $RI_k, (i \leq k \leq n)$ is comprised of a trust value and level pair (tv_k, tl_k) for a single evaluated trusted agent. Each trust value tv_k is mapped to a single trust level tl_k, such that $(0 \leq tv_k, tl_k \leq 1)$.

ReportItem

$tv! : fuzzy\_value$

$tl! : tv! \rightarrow fuzzy\_value$

$rules : \mathbb{P}\, Rule$

Definition 6.1.1: Rule A *Rule* (R) is a breakdown of each decision rule and the degree of participation within that rule. For example, suppose rule ID=3 has a degree value of 0.27, this implies 27% participation in fuzzyset ID=3.

Rule

$ruleId! : ID$

$degree! : ruleId \rightarrow fuzzy\_value$

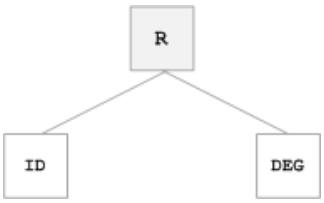


Figure 3.21: Rule building blocks (R = Rule, ID = ID, DEG = Degree)

3.4 Conclusions

The objective of this chapter was to describe the synergistic reputation-policy trust model including merits, considerations and limitations. This was achieved by discussing the characteristics of the model whilst reflecting on its value-added functionalities and comparing them to the state of the art. This aspect also involved listing global considerations, constraints and assumptions. In addition, the structure of the model was elaborately described including its internal artifacts and process interactions. In conclusion, the main outputs of this chapter are the following:

- Proposition of a *synergistic* reputation-policy based paradigm which combines elements of reputation-based and policy-based trust models; that is subjectively stipulating reputation evaluation criteria using policy assertions.
- Promotion of an *exoteric* reputation-based trust model which endeavours active participation in the trust and reputation evaluation processes; that is a trust model in which the metrics for trust and reputation are created, selected and managed externally utilising trust decision strategies which reflect on the submitted job QoS requirements, the global trust context and risk aptitude.
- Consolidation of an evaluation model based on a multi-faceted discrete trust model and a fuzzy logic based decision model thus tightly coupling the notions of trust and risk and forming a consolidated *fuzzy-discrete* trust model. The conjunction of this chapter outputs constitute as the first research contribution: *RC1: Synergistic Reputation-Policy Based Trust Model.*

Chapter 4

Grid Reputation-Policy Based TMS Architecture & Querying Mechanism

"Reality is merely an illusion, albeit a very persistent one."

Albert Einstein

4.1 Introduction

The purpose of this chapter is to introduce the *Grid Reputation-Policy Based Trust Management Service* (GREPTrust) [78]. The main objective of the GREPTrust service is to provide a solid reference implementation for the synergistic reputation-policy based trust model. This involves defining a service architecture which incorporates the various artifacts of the model into a single, self-contained Grid reputation-based trust management service. This definition is supported by an analysis of the architecture components (RC2) placing additional emphasis on the Grid resource querying mechanism (RC3) and the VO aggregated querying mechanism (RC4).

Therefore, this chapter is comprised of two sections - Grid Reputation-Policy Based TMS architecture and the Grid Reputation-Policy Based Querying Mechanism. The first section provides an overview of the service architecture including its three subsystems (client, service and data), its scope within service-oriented environments and the main characteristics which differ GREPTrust from existing Grid reputation-based TMS. The second section elaborately discusses the reputation querying mechanism provided by the GREPTrust architecture including querying support for a single or a set of resources as well as for aggregated reputation queries.

4.2 Grid Reputation-Policy Based TMS Architecture

4.2.1 Overview

Introduction

The purpose of GREPTrust is to provide a solid reference implementation for the reputation-policy based trust model. GREPTrust is required to support VOs during the formation and operation phases, that is - during the negotiation period with the VO candidates regarding their participation in the VO as well as during the main life-cycle execution period. During the VO formation phase GREPTrust could be used to assist in screening the participating parties to ensure that they would be able to fulfil their service level agreements. GREPTrust provides three major functions for the participating entities: capturing reputation-policy requirements, computing resources trustworthiness and evaluating resources after transactions.

GREPTrust properties can be summarised in Table 4.1. It is a non-centralised trust model as trust data is distributed between Grid clients and the reputation algorithm. The Grid clients submit TDS files containing the trust metrics to the reputation algorithm which exploits opinion matrices which gather their data from the ratings feedback records stored in the reputation-policy data store. The type of the model is a *synergistic* reputation-policy as it leverages policy assertions for evaluation resources reputation. Therefore, the nature of the reputation algorithm is *exoteric*, as it allows external manipulation to the trust and reputation evaluation process using subjective statements which are injected to the correlation process. This property stands in contrast to all existing Grid reputation-based trust management systems. The classification of this solution is *fuzzy-discrete*. Trust metrics normalise trust values and represent them as a fuzzy value ranging between 0 and 1. Evaluation feedback is supplied as continuous values which are calculated based on the discrepancies between the SLA agreement and actual quality of service provided.

#	Property	Value
1.	System	GREPTrust
2.	Classification	Deterministic-Approximative » Fuzzy-Discrete
3.	Centralised data	No
4.	Trust metrics	[0,1]
5.	Trust aggregation	Yes
6.	Multi QoS aspects	Yes
7.	Esoteric reputation querying	No
8.	QoS/SLA negotiation	QoS negotiation via policy assertions
9.	Type of feedback	Continuous

Table 4.1: GREPTrust properties summary

Functional Requirements

GREPTrust is required to seamlessly integrate into the service provider-consumer model of SOA-based architectures such as Web services and service oriented grids. With consideration to Grid computing environments, GREPTrust is argued to be suited for providing advanced services as part of a core Grid middleware. It is expected to address three requirements: (i) trust evaluation of a single or a set of resources, (ii) trust aggregation based on an organisational and VO basis and (iii) inference of the trustworthiness of a single resource based on the global context (i.e. either via the trust of the organisation the resource belongs to or the entire VO).

For example, a typical Grid reputation-policy based scenario involves a resource broker acquiring information regarding the availability of computing resources utilising an information service such as the MDS [89] or R-GMA [88, 90]. Resources may advertise their capabilities through SLA contracts and publish them to the information service so that they can be queried by the resource broker. Each SLA contract contains one or more resource-guaranteed QoS parameters. It may specify the levels of availability, serviceability, performance, operation, or other attributes of the service like billing and even penalties in the case of violation of the SLA.

Let Φ define the universal set of resource-guaranteed QoS parameters. A job submitted to the resource broker is manifested by a job description file, which contains a list of parameters describing the job. In the context of this thesis, two additional parameters are introduced: (i) store a reference to a Trust Decision Strategy T that is associated with the type of the submitted job and (ii) a threshold value v indicating the minimum expected resource trust level.

The evaluation model ramification of the TDS contains a permutation of client-expected QoS parameters. Let Θ define the set of client-expected QoS parameters. The resource broker can filter the list of potential resources R_Θ based on $\Theta \subseteq \Phi$, such that the list of client-expected QoS parameters is a subset of the resource-guaranteed ones. T and R_Θ are submitted to the reputation-policy trust model which processes the request and returns a trustworthiness report for each resource $r \in R_\Theta$. Each resource trust level r_v is compared with the threshold v using $(\forall r \mid r \subseteq R_\Theta \rightarrow r_v \geq v)$ in order to generate the final set of resources for selection.

After transaction completion, the Grid client compares the actual level of service provided with the one guaranteed by the SLA contract. Any noted differences are recorded and deducted from the total score. The Grid client uses the same TDS used for selecting the resource to indicate which factors to provide ratings for. For example, if a TDS contains availability and reliability as evaluation factors, the Grid client uses these factors as comparison indicators. Once the comparison is completed, the Grid client submits a feedback report which contains for each resource, a break down of each factor and the feedback value. GREPTrust collects these values and

stores them in the Reputation-Policy Data Store.

Figure 4.1 illustrates an example of a Grid middleware architecture utilising GREPTrust. A job submitted via a Grid portal reaches a resource broker (1). The resource broker contacts a Grid Information Service (GIS) in order to retrieve a list of available resources and their published resource-guaranteed QoS capabilities (2). The resource broker contacts the TDS repository to retrieve the desired TDS for the given job including client-expected QoS capabilities (3). The resource broker decides on the final list of resources, constructs a reputation-policy query and submits the query to GREPTrust (4) which evaluates the query and then returns a reputation-policy report. The resource broker evaluates the report and decides on the final list of resources to utilise for the job computation (5). During the execution of the job, a Grid Monitoring Service (GMS) monitors the performance of the selected resources (6) and once completed - it provides feedback ratings (7).

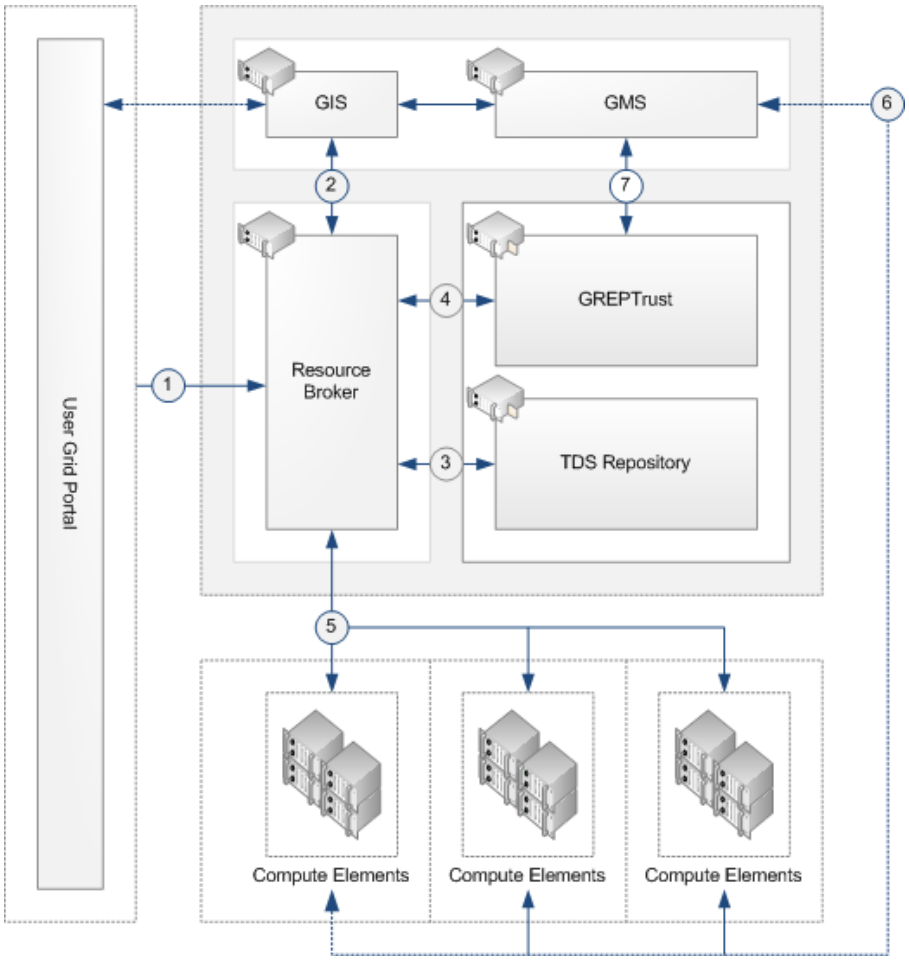


Figure 4.1: GREPTrust Middleware solution

4.2.2 System Components

Overview

Figure 4.2 illustrates the GREPTrust architecture, which is comprised of three underlying subsystems: *Client system* (Grid Client, TDS Repository), *Service system* (Reputation-Policy Service Façade, Querying Manager, Feedback Manager and Admin Manager) and *Data system* (Reputation-Policy Data Store).

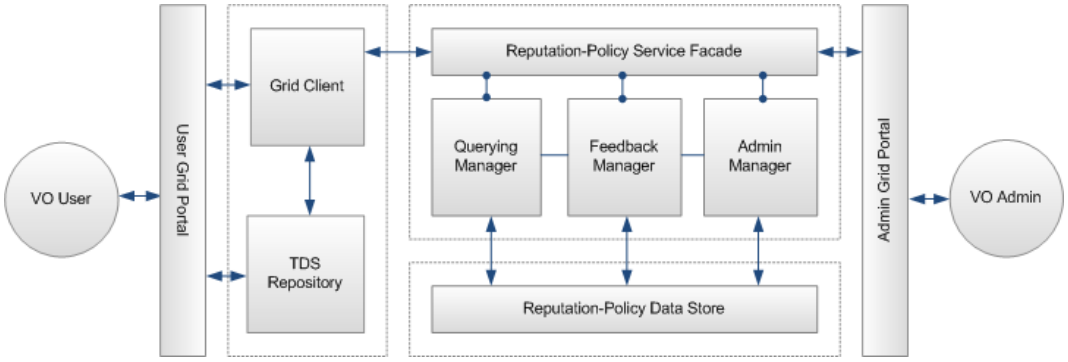


Figure 4.2: GREPTrust Management Service architecture

Client System

The client system enables clients (e.g. resource brokers, monitoring toolkits) to select predefined trust decision strategies from the TDS Repository by specifying an ID which identifies the strategy to be processed by the reputation algorithm. It is important to note that prior to strategy selection, the client should analyse on the current job requirements, the available ratings feedback information and the VO characteristics and selects the most suitable strategy used for resource evaluation.

Since this *strategy selection* algorithm is outside the scope of this research, it has been identified as a possible direction for future work. At the current state, however, it is assumed that an analysis has been conducted by the Grid client prior to selecting the strategy from the TDS repository. Once a strategy is selected, a reference to the strategy is submitted to a Grid client, which obtains the strategy from the repository and appends it to a reputation-policy query. In addition, the query also contains a set of resources to be evaluated and context information.

Service System

The Grid client submits the query on behalf of the VO user to the reputation-policy service façade, which incorporates the façade design pattern thus providing simplified interface to GREPTrust internal components. The reputation-policy service façade contacts the querying manager to calculate the trust values based on the

given resources and the strategy instructions. The querying manager calculates the trust values by contacting the Reputation-Policy Data Store (RPDS) for obtaining historical reputation feedbacks and manipulating these records with the evaluation criteria set in the TDS. The trust values are then passed through a Decision Rules Engine (DRE) which uses the decision rules supplied inside the Decision Model (DM) of the TDS to map each trust value into trust level.

The final step involves generating a Reputation-Policy Report (RPR), which is essentially a structured report containing each evaluated resource and its associated trust level. The trust level indicates the amount of trust a Grid client should have in the evaluated resource based on the given TDS. It is important to note that GREPTrust leaves the rank/threshold based functions to be decided upon by the Grid client. For example, if the Grid client defines a threshold value of 0.8, a resource which obtains a trust level of 0.9 it will be valid for selection whereas a resource which obtained a trust level of 0.7 will be rejected.

In addition to resource querying management, GREPTrust is designed to allow the provision of resource evaluation feedbacks after a job has been completed. This is available through the Feedback Manager (FM) component. Grid clients are required to provide reputation feedbacks for each of the utilised Grid resources so that feedback could be used for future resource evaluations. The purpose of the Admin Manager (AM) is to manage the historical reputation feedback data available inside the RPDS. The underlying data model is described in the following section.

Data System

GREPTrust relies on a well defined relational data model for storing and manipulating historical rating feedbacks (Figure 4.3). It contains five different entities:

- *Client* - The client entity stores information regarding Grid clients. This entity is important for identifying the Grid client making the reputation evaluation request as well as for calculating reputation values when requiring to differentiate between the current Grid client experience and reputation recommendations made by peer Grid clients.

- *Resource* - The resource entity stores information regarding Grid resources. This information is used when Grid clients select to evaluate a single resource, a collection of resources or the entire VO.

- *Execution* - The execution entity stores information regarding job executions (e.g. execution ID, client ID, resource ID and completion datetime). This is the most important entity in the model as it intermediates Grid clients and

resources and keeps track of the historical executions. Each execution may contain one or more feedbacks.

– *Feedback* - The feedback entity stores information regarding feedbacks supplied by Grid clients. A feedback is a segregated piece of rating information supplied and stored as part of an execution and it is always associated with a single opinion type. For example, consider an execution between Grid client *A* and computing resource *B*. Upon execution completion, the execution entity will store the execution details and the feedback entity will store any feedback *A* supplies regarding *B*. If *A* used the opinions availability and reliability as reputation evaluation criteria, then two records will be added to the feedback entity, each of each contains feedback value ranging between 0 and 1.

– *Opinion* - The opinion entity stores information regarding available opinion types supported by the VO (e.g. availability, reliability and etc). When a reputation feedback entry is recorded, it is always associated to an opinion which is defined in the opinion entity and linked to the feedback entity.

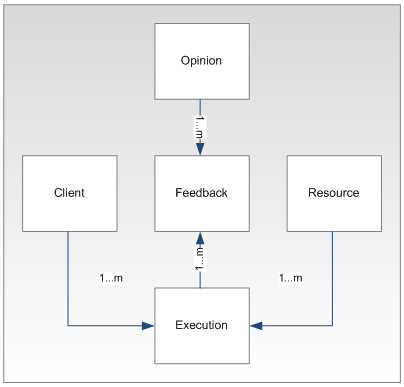


Figure 4.3: RPDS - Reputation-Policy Data Store entity-relationship diagram

4.3 Grid Reputation-Policy Based Querying Mechanism

4.3.1 Overview

The Reputation-Policy Based Querying Mechanism is responsible for the reputation querying facilities in GREPTrust. There are three types of supported reputation queries: (i) reputation queries for a single or a set of resources (ii) aggregated reputation queries which allow querying the reputation of an organisation based on the reputation of each of its contributed resources as well as querying the reputation of an entire VO based on the reputation of its underlying organisations; and (iii) Grid resource inference, which allows inferring about the trustworthiness of a single

resource based on the global context. These features address the aforementioned requirements for reputation management in Grid computing. Section .2.3 in appendix .2 contains sample outputs for each of the supported types of reputation-policy queries resulting from invoking the reputation-policy querying mechanism.

4.3.2 Grid Resource Reputation Querying Mechanism

Overview

The Grid resource reputation querying mechanism is available through the Querying Manager (QM) component of GREPTrust. This component is directly in charge of Grid resource querying facilities. In addition, multiple instances of the QM can be utilised for performing aggregated reputation queries intended for discovering the reputation of organisations based on their individual resources they contribute. Figure 4.4 illustrates the architecture of the Query Manager.

The querying mechanism enables Grid clients to evaluate resources based on the reputation evaluation criteria they supply to the Query Manager. Grid clients are required to submit a Reputation-Policy Query (RPQ) to the Query Manager, which contains the trust decision strategy as well as context information such as client ID, the IDs of the resources to evaluate, the cutoff date time (the start date of which to gather the feedback data) and the trust decay function which specifies the rate of decay for historical rating feedbacks. The QM processes the RPQ and generates a Reputation-Policy Report (RPR) and sends it back to the Grid clients.

Appendix .2.1 contains several class diagrams illustrating the structure the Querying Manager (QM). This includes RPQ, TDS, EM, DM, RPR, CP, CMP diagrams.

Algorithm 1 describes the three steps required for the Reputation Algorithm (RA) to generate an RPR from RPQ: (i) Processing the TDS Evaluation Model (lines 16-18), (ii) Processing the TDS Decision Model (lines 19-36); and (iii) Generating a Reputation-Policy Report (lines 37-38). Appendixes .2.2 and .2.2 include sample TDS and RPR files correspondingly. In line 16, a TDS is obtained by extracting it from the RPQ and deserialising into an object model using the TDSSerializer. In line 17, a new instance of the Correlation Process (CP) is created supplying it with the Evaluation Model (EM) and Context Data (CD). In line 18, the *Process* method is called on the CP which returns a Resource Value Table (RVT). The RVT is defined as a hashtable consisting of resource identifier and trust value pairs.

In line 19, the DM is extracted from the TDS. If the DM was defined in the TDS then a new instance of the Decision Rule Engine (DRE) is created (lines 20-22). The

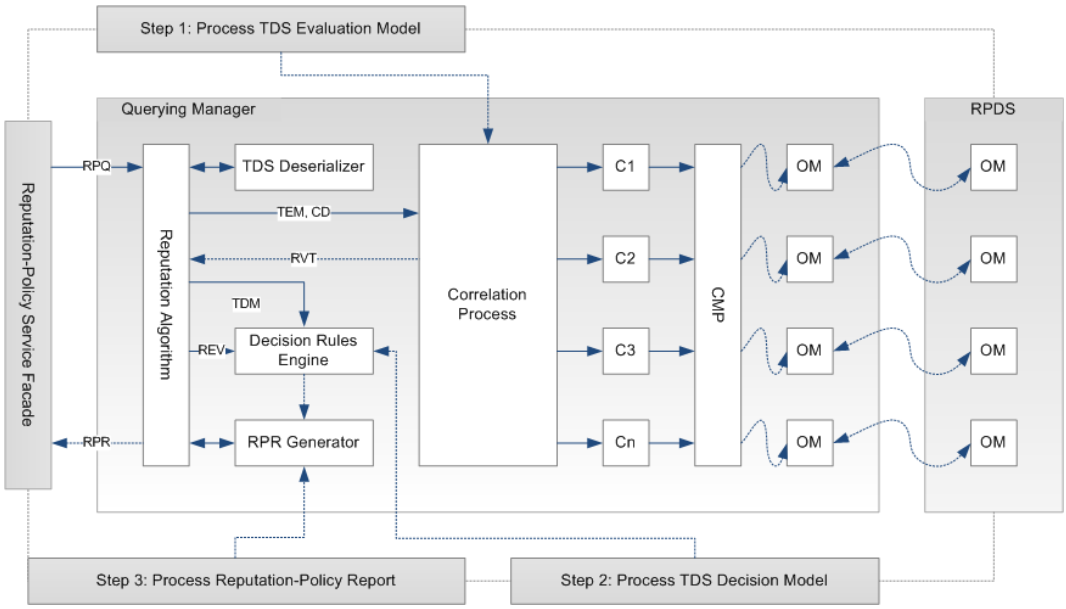


Figure 4.4: Query Manager architecture

DRE is a wrapper around a Fuzzy Inference System (FIS) which enables mappings of trust values onto trust levels. In line 23, a new instance of a *Report* entity is created used for generating the Reputation-Policy Report (RPR).

In lines 24-36, the Reputation Algorithm (RA) iterates through the RVT. In line 25, a new instance of a *Resource* entity is created supplying it with the resource key *ReK* and the computed trust value *ReV*. In line 27, each resource trust value *ReV* is passed as an argument to the *Process* method. The *Process* method returns a Rule Table (*RuT*). The *RuT* is defined as a hashtable in which the *TRUST_LEVEL* key points to the inferred trust level whilst the remaining keys are the rule identifiers point to the evaluated trust value degree of membership in each rule.

In lines 28-34, the Reputation Algorithm (RA) iterates through the *RuT*. If the key is *TRUST_LEVEL* then the trust level *RuV* is assigned to the Resource entity (line 30). Otherwise, a Rule entity is appended to the Resource entity, consisting of the rule ID *RuK* and the degree of membership *RuV*.

In line 37, a new instance of a Reputation-Policy Report (RPR) is created by supplying it with the Report entity. The RPR is returned to the calling Grid client (line 38). The following subsections elaborate on processing each of these steps.

Algorithm 1: RA::Process(RPQ) algorithm

```
1: RPQ: reputation-policy query
2: CP: correlation process
3: RVT: resource value table
4: CD: context data
5: EM: evaluation model
6: DM: decision model
7: DRE: decision rule engine
8: Rep: report entity
9: Res: resource entity
10: ReK: resource key
11: ReV: resource value
12: ReT: rule table
13: RuK: rule key
14: RuV: rule value
15: RPR: reputation-policy report
16: TDS ← TDSDeserializer(TDS(RPQ))
17: CP ← Create(EM(TDS), CD(RPQ))
18: RVT ← Process(CP)
19: DM ← DM(TDS)
20: if DM ≠ null then
21:    DRE ← Create(DM)
22: end if
23: Rep ← Create()
24: for all (ReK, ReV) ∈ RVT do
25:    Res ← Create(ReK, ReV)
26:    if DM ≠ null then
27:       RuT ← Process(DRE, ReV)
28:       for all (RuK, RuV) ∈ RuT do
29:          if RuK = TRUST_LEVEL then
30:             AssignLevel(Res, RuV)
31:             continue
32:          end if
33:          AddRule(Res, RuK, RuV)
34:       end for
35:    end if
36: end for
37: RPR ← RPRGenerator(Rep)
38: return RPR
```

TDS Evaluation Model Processing

Overview The first step in processing the RPQ involves processing the TDS Evaluation Model. During this step the Reputation-Algorithm (RA) utilises the TDSDeserializer to extract the trust decision strategy out of the Reputation-Policy Query (Algorithm 1, line 16). The purpose of the TDSDeserializer is to convert the input TDS (marshalled as an XML string) into an object domain model. Consecutively, the RA creates a new instance of the Correlation Process (CP) passing it the Evaluation Model (EM) and Context Data (CD) as arguments (Algorithm 1, line 17). Finally, the Process method is called on the CP which returns a Resource Value Table (RVT) entity once the method completed (Algorithm 1, line 18).

The following displays an example of the EM, including an evaluation for two opinions (O_1, O_2) where O_1 has 0.7 weight ratio and O_2 has 0.3 weight ratio. Both opinions define experience and reputation as trust sources - both 0.5 weight ratio.

```
1  <TrustEvaluationModel>
2      <Opinions>
3        <Opinion Type="1" Weight="0.7">
4          <Sources>
5            <Source Type="Experience" Weight="0.5"/>
6            <Source Type="Reputation" Weight="0.5"/>
7          </Sources>
8        </Opinion>
9        <Opinion Type="2" Weight="0.3">
10         <Sources>
11           <Source Type="Experience" Weight="0.5"/>
12           <Source Type="Reputation" Weight="0.5"/>
13         </Sources>
14       </Opinion>
15     </Opinions>
16 </TrustEvaluationModel>
```

Correlation Process Algorithm 2 further elaborates on the CP Process method. For each opinion entity O in the Opinion List (OL) of the Evaluation Model (EM), the correlation process spawns a new child process - Child Correlation Process (CCP) and supplies it with the opinion and Context Data (CD) as arguments (Algorithm 2, line 10). The *Process* method is called on each CCP in a separate thread (Algorithm 2, line 11). All CCPs are granted synchronised access to a shared entity - Opinion Accumulator (OA). The purpose of the OA is to intermediately store $1:M$ mappings of resource identifiers and weighted values, one for each evaluated opinion. Once all CCPs completed, the OA is ready to be processes into a Resource Value Table (RVT) utilising the *Summary* method (Algorithm 2, line 13).

Algorithm 2: CP::Process() algorithm

```
1: EM: evaluation model
2: OL: opinion list
3: CD: context data
4: O: opinion entity
5: OA: opinion accumulator
6: CCP: child correlation process
7: RVT: resource value table
8: OL ← OL(EM);
9: for all O ∈ OL do
10:    spawn CCP ← Create(O, CD)
11:    Process(CCP)
12:    sync
13:    RVT ← Summary(OA)
14: end for
```

Child Correlation Process Algorithm 3 elaborates on the CCP *Process* function. This algorithm describes the processing of an arbitrary opinion O_k within the OL (where $1 \leq k \leq |\, OL \,|$). The Customised Matrices Pool (CMP) is supplied with opinion O_k and Context Data (CD) and returns an $m \times n$ Opinion Matrix (OM) for opinion O_k - i.e. $OM(O_k)$ where m refers to the number of evaluating clients and n refers to the number of evaluated resources (Algorithm 3, line 6). An entry in the matrix $OM(O_k)_{[i,j]}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ defines feedback value client i has accumulated for resource j. This feedback value is based on formula 3.1 in chapter 3.

Algorithm 3: CCP::Process() algorithm

```
1: CMP : customised matrices pool
2: CD : context data
3: OM(O_k) : opinion matrix
4: O_k : opinion
5: RVT(O_k) : resource value table for O_k
6: OM(O_k) ← CMP(O_k, CD)
7: RVT(O_k) ← Solve(OM(O_k))
8: while at synchronised state do
9:    Put(OA, O_k, RVT(O_k))
10: end while
```

The *Solve* function is called on $OM(O_k)$ which returns a Resource Value Table (RVT) for opinion O_k - i.e. $RVT(O_k)$. $RVT(O_k)$ is defined as a hash table consisting of the set of evaluated resources $\{R_1, \ldots, R_n\}$ as keys and the set $\left\{\overline{V(R_1)_{O_k}}, \ldots, \overline{V(R_n)_{O_k}}\right\}$ as associated opinion-level trust values. This is denoted by formula 4.1:

$$RVT(O_k) = \begin{cases} R_1 \mapsto \overline{V(R_1)_{O_k}} \Leftrightarrow \omega_e V(C_1) + \omega_r \frac{\sum_{i=2}^{|CT_1|} V(C_i)}{|CT_1|} \\ R_2 \mapsto \overline{V(R_2)_{O_k}} \Leftrightarrow \omega_e V(C_1) + \omega_r \frac{\sum_{i=2}^{|CT_2|} V(C_i)}{|CT_2|} \\ R_3 \mapsto \overline{V(R_3)_{O_k}} \Leftrightarrow \omega_e V(C_1) + \omega_r \frac{\sum_{i=2}^{|CT_3|} V(C_i)}{|CT_3|} \\ \vdots \\ R_n \mapsto \overline{V(R_n)_{O_k}} \Leftrightarrow \omega_e V(C_1) + \omega_r \frac{\sum_{i=2}^{|CT_n|} V(C_i)}{|CT_n|} \end{cases} \tag{4.1}$$

The trust value of a resource R_r $(1 \leq r \leq n)$ is defined as a weighted mean comprising of two components: (i) the requesting client feedback value $V(C_1)$ multiplied by the stipulated experience weight factor ω_e ;and (ii) the average of feedback values provided by all the other clients $\frac{\sum_{i=2}^{|CT|} V(C_i)}{|CT_i|}$ (where $\mid CT \mid = m$) multiplied by the stipulated reputation weight factor ω_r. This is denoted by formula 4.2:

$$\overline{V(R_r)_{O_k}} = \omega_e V(C_1) + \omega_r \frac{\sum_{i=2}^{|CT_r|} V(C_i)}{\mid CT_r \mid}, (2 \leq \mid CT_r \mid, \omega_e + \omega_r = 1) \tag{4.2}$$

Opinion Matrix Algorithm 4 elaborates on the OM *Solve* function. The principle behind this function is to transpose the opinion matrix $OM(O_k)$, such that each pair $(r, ct) \in \{R_1 \mapsto CT_1, \ldots, R_n \mapsto CT_n\}$ defines mappings between each evaluated resource r and client table ct which is comprised of grouping each client and feedback value. This serves the advantage of calculating the trust value of each resource r concurrently. This is denoted by formula 4.3:

$$\overbrace{\begin{array}{l} R_1 \mapsto \begin{cases} \overbrace{C_1 \mapsto V_1}^{CT_1} \\ C_2 \mapsto V_2 \\ C_3 \mapsto V_3 \\ \vdots \\ C_m \mapsto V_m \end{cases} \cdots R_2 \mapsto \begin{cases} \overbrace{C_1 \mapsto V_1}^{CT_2} \\ C_2 \mapsto V_2 \\ C_3 \mapsto V_3 \\ \vdots \\ C_m \mapsto V_m \end{cases} \cdots R_3 \mapsto \begin{cases} \overbrace{C_1 \mapsto V_1}^{CT_3} \\ C_2 \mapsto V_2 \\ C_3 \mapsto V_3 \\ \vdots \\ C_m \mapsto V_m \end{cases} \cdots R_n \mapsto \begin{cases} \overbrace{C_1 \mapsto V_1}^{CT_n} \\ C_2 \mapsto V_2 \\ C_3 \mapsto V_3 \\ \vdots \\ C_m \mapsto V_m \end{cases} \end{array}}^{RT(O_k)=} \tag{4.3}$$

The transposed matrix is stored in a Resource Table ($RT(O_k)$) entity (Algorithm 4, line 15). The value $V(R_r)_{O_K}$ represents a computed trust value for resource r and opinion O_k. It can contain one of the following options: (i) -1, which is a default value in case neither experience nor feedback values recorded (Algorithm 4, line 20); (ii) mean of reputation values, in case no experience is recorded (Algorithm 4, line 22); (iii) the experience value $V(C_1)$ in case no reputation feedbacks are recorded

(Algorithm 4, line 26) and finally (iv) - experience and reputation (Algorithm 4, line 28). The return value of the *Solve* function is $RVT(O_k)$ hash table.

Algorithm 4: OM::Solve() algorithm

1: CMP : customised matrices pool
2: $OM(O_k)$: opinion matrix for opinion O_k
3: O_k : opinion
4: R_r : arbitrary resource r
5: CID : client ID
6: CT_r : client table associated to R_r
7: $RT(O_k)$: resource table (transposed opinion matrix)
8: $| CT_r |$: size of client table
9: $| RT |$: size of resource table
10: $V(C_1)$: trust value supplied by the client
11: $V(R_r)_{O_K}$: computed trust value for resource r and opinion O_k
12: ω_e : experience weight factor
13: ω_r : reputation weight factor
14: $RVT(O_k)$: resource value table for opinion O_k
15: $RT(O_k) \leftarrow OM(O_k)^T$
16: **for parallel** R_r such that $1 \leq r \leq | RT(O_k) |$ **do**
17: $V(C_1) \leftarrow Remove(CT_r[CID])$
18: **if** $V(C_1) = -1$ **then**
19: **if** $| CT_r | = 0$ **then**
20: $V(R_r)_{O_K} \leftarrow V(C_1)$ {neither experience nor reputation}
21: **else**
22: $V(R_r)_{O_K} \leftarrow \frac{\sum_{i=1}^{|CT_r|} V(C_i)}{|CT_r|}$ {reputation only}
23: **end if**
24: **else**
25: **if** $| CT_r | = 0$ **then**
26: $V(R_r)_{O_K} \leftarrow V(C_1)$ {experience only}
27: **else**
28: $V(R_i)_{O_K} \leftarrow \left[\omega_e V(C_1) + \omega_r \frac{\sum_{i=1}^{|CT_i|} V(C_i)}{|CT_i|} \right]$ {experience and reputation}
29: **end if**
30: **end if**
31: $RVT(O_k)[(R_r, V(R_r)_{O_K})]$
32: **end for**
33: **return** $RVT(O_k)$

The $RVT(O_k)$ hash table is supplied together with O_k as arguments to *Put* function (Algorithm 3 line 9). This function iterates through the elements of $RVT(O_k)$ and each trust value is multiplied with the stipulated opinion weight ω_{O_k} $(0 \leq \omega_{O_k} \leq 1)$ and stored in a Value List (VL) hash table:

for all $(R_r, V_{O_k}) \in RVT(O_k)$ **do**
  $VL[(R_r, \omega_{O_k} \times V_{O_k})]$
end for

Opinion Accumulator As aforementioned, the Opinion Accumulator (OA) stores $1:M$ mappings of resource identifiers and weighted trust values. It is a hash table where the set of keys $\{R_1, \ldots, R_n\}$ is named Resource List (RL) and the set of values - Value List (VL) contains an associated set of weighted values $\left\{\omega_{O_1} V_{O_1}, \ldots, \omega_{O_{|OL|}} V_{O_{|OL|}}\right\}$, one for each resource $R_r \in RL$. This is denoted by formula 4.4:

$$OA = \begin{cases} \overbrace{R_1}^{RL} \mapsto \overbrace{\left\{\omega_{O_1} V_{O_1}, \omega_{O_2} V_{O_2}, \ldots, \omega_{O_{|OL|}} V_{O_{|OL|}}\right\}}^{VL} \\ R_2 \mapsto \left\{\omega_{O_1} V_{O_1}, \omega_{O_2} V_{O_2}, \ldots, \omega_{O_{|OL|}} V_{O_{|OL|}}\right\} \\ R_3 \mapsto \left\{\omega_{O_1} V_{O_1}, \omega_{O_2} V_{O_2}, \ldots, \omega_{O_{|OL|}} V_{O_{|OL|}}\right\} \\ \vdots \\ R_n \mapsto \left\{\omega_{O_1} V_{O_1}, \omega_{O_2} V_{O_2}, \ldots, \omega_{O_{|OL|}} V_{O_{|OL|}}\right\} \end{cases} \tag{4.4}$$

Algorithm 5 elaborates on the *Summary* function (Algorithm 2, line 13). The purpose of this function is to generate the final RVT entity from the OA.

Algorithm 5: OA::Summary() algorithm

```
1:  R: resource
2:  TV(R): trust value for resource R
3:  VL: value list
4:  OA: opinion accumulator
5:  RK: resource key
6:  WV: weighted value
7:  RVT: resource value table
8:  for all (R, VL) ∈ OA do
9:     TV(R) ← 0
10:    for all (RK, WV) ∈ VL do
11:       if RK = R then
12:          TV(R) ← TV(R) + WV
13:       end if
14:    end for
15:    RVT[(R, TV(R)]
16: end for
17: return RVT
```

The final RVT entity is a hash table consisting of the keys $\{R_1,\ldots,R_n\}$ and a set of values $\left\{\overline{TV(R_1)},\ldots,\overline{TV(R_n)}\right\}$. This is denoted by formula 4.5:

$$RVT=\begin{cases} R_1 \mapsto \overline{TV(R_1)} \Leftrightarrow \sum_{n=1}^{|OL|} \omega_{O_n} V_{O_n} \\ R_2 \mapsto \overline{TV(R_2)} \Leftrightarrow \sum_{n=1}^{|OL|} \omega_{O_n} V_{O_n} \\ R_3 \mapsto \overline{TV(R_3)} \Leftrightarrow \sum_{n=1}^{|OL|} \omega_{O_n} V_{O_n} \\ \vdots \\ R_n \mapsto \overline{TV(R_n)} \Leftrightarrow \sum_{n=1}^{|OL|} \omega_{O_n} V_{O_n} \end{cases} \tag{4.5}$$

Each resource R_i is mapped to summation of the set of weighted opinion-level trust values $\left\{\omega_{O_1}V_{O_1},\ldots,\omega_{O_{|OL|}}V_{O_{|OL|}}\right\}$ in VL, yielding the overall trust value $\overline{TV(R_i)}$. This is denoted by formula 4.6:

$$\overline{TV(R_i)}=\omega_{O_1}V_{O_1}+\omega_{O_2}V_{O_2}+\ldots+\omega_{O_{|OL|}}V_{O_{|OL|}}=\sum_{n=1}^{|OL|}\omega_{O_n}V_{O_n} \tag{4.6}$$

Since $V_{O_n}=\overline{V(R_1)_{O_n}}$, formula 4.2 can be used to represent overall trust value for resource R_r. This is denoted by formula 4.7:

$$\overline{TV(R_i)}=\sum_{o\in OL}\omega_{O_o}\left[\omega_e V(C_1)+\omega_r\frac{\sum_{i=2}^{|CT_i|}V(C_i)}{\mid CT_i \mid}\right],\left(\sum_{o\in OL}\omega_{O_o}=1\right) \tag{4.7}$$

The final RVT entity is returned by the OA *Summary* function in (Algorithm 5, line 17) to the CP *Process* function in (Algorithm 2, line 13) which eventually returns to the RA *Process* function in (Algorithm 1, line 18). This completes the required steps for the processing the evaluation model.

Figure 4.5 illustrates the Correlation Process (CP) architecture and summarises its underlying processing steps:

1. Spawning a Child Correlation Process: $(CCP(O_k,CD)$.
2. Contacting the CMP and obtaining an associated $OM(O_k,CD)$.
3. Generating an opinion-level $RVT(O_k)$.
4. Constructing the Opinion Accumulator OA from the underlying opinion level $\{RVT(O_1),\ldots,RVT(O_n)\}$.
5. Producing the final RVT from OA.

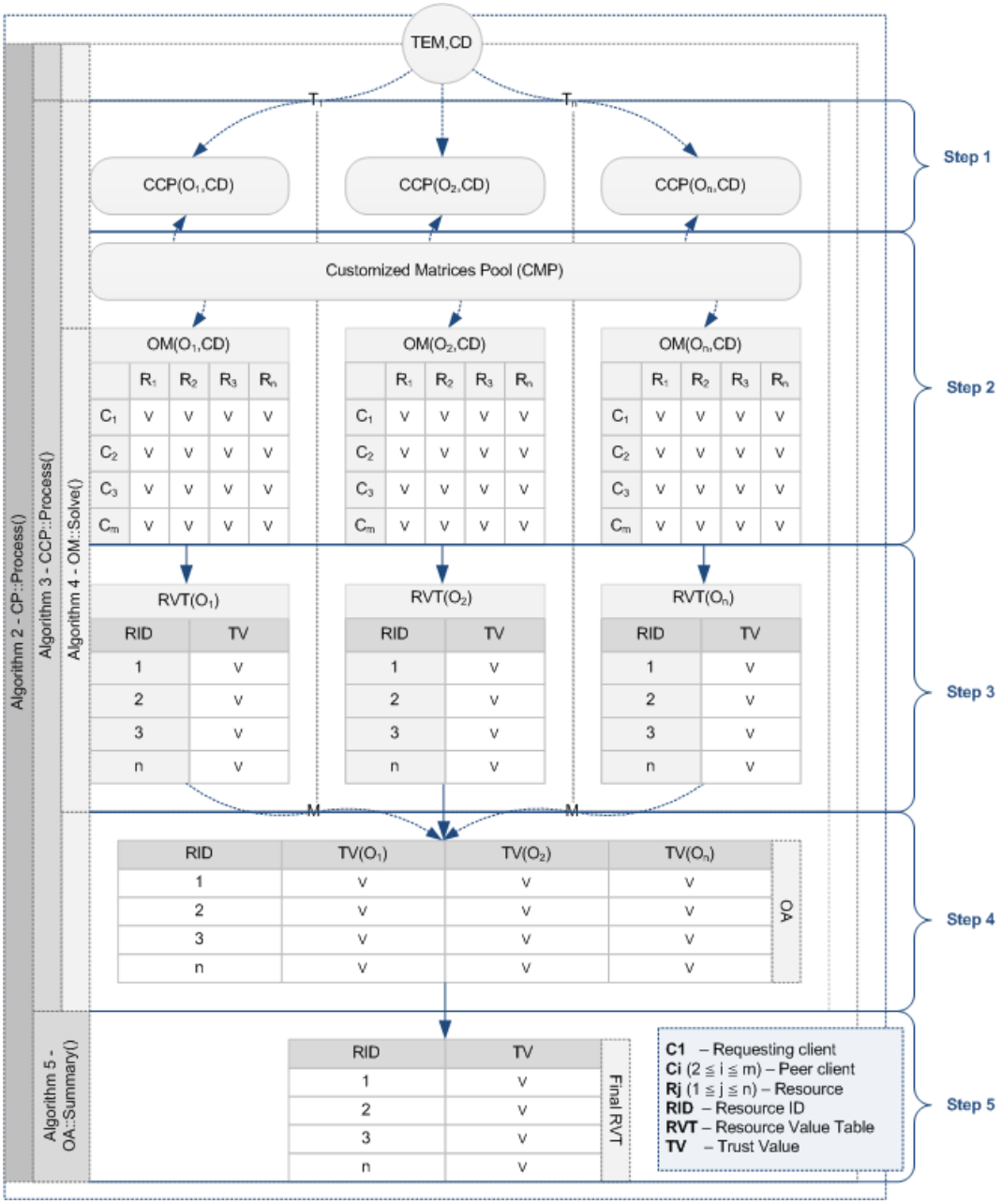


Figure 4.5: Correlation Process Architecture

TDS Decision Model Processing

Overview The second step in processing the RPQ involves processing the Decision Model (DM). The purpose of this step is to map the trust value $\overline{TV(R_i)}$ onto a trust level $\overline{TL(R_i)}$ using fuzzy inference function Φ, denoted by formula 4.8:

$$\overline{TL(R_i)} = \Phi(\overline{TV(R_i)}), (0 \leq TL(R_i) \leq 1) \tag{4.8}$$

During this step, the Reputation-Algorithm (RA) obtains the DM from the TDS and then creates a new instance of Decision Rules Engine (DRE) (Algorithm 1, lines 19-22). The DRE is essentially a wrapper object for the JFuzzyLogic [131] Fuzzy Inference System (FIS) library. When a new instance of the FIS is created, the DM is passed as an argument to the DRE constructor and then converted into Fuzzy Control Language (FCL [82]) language constructs (Algorithm 1, line 21).

Consecutively, the RA iterates through the RVT (Algorithm 1, lines 24-36). Each resource trust value *ReV* is passed as an argument to the process method (Algorithm 1, lines 27). Internally, the DRE obtains the Fuzzy Rule Set (FRS) from the FIS. The FRS defines a set of IF-THEN mapping rules, obtained from the DM. The DRE sets the trust value variable inside the FRS and passes in the trust value and eventually calls the *Evaluate* function on the FRS. The outcome of the TDS decision model Processing is a a Rule Table *RuT*, in which the *TRUST_LEVEL* key points to the inferred trust level whilst the remaining keys are the rule identifiers pointing to the evaluated trust value degree of membership in each rule. Figure 4.6 illustrates the DRE architecture. It is comprised of three components: *fuzzification*, *defuzzification* and *rule-based inference engine*, described in the following sections.

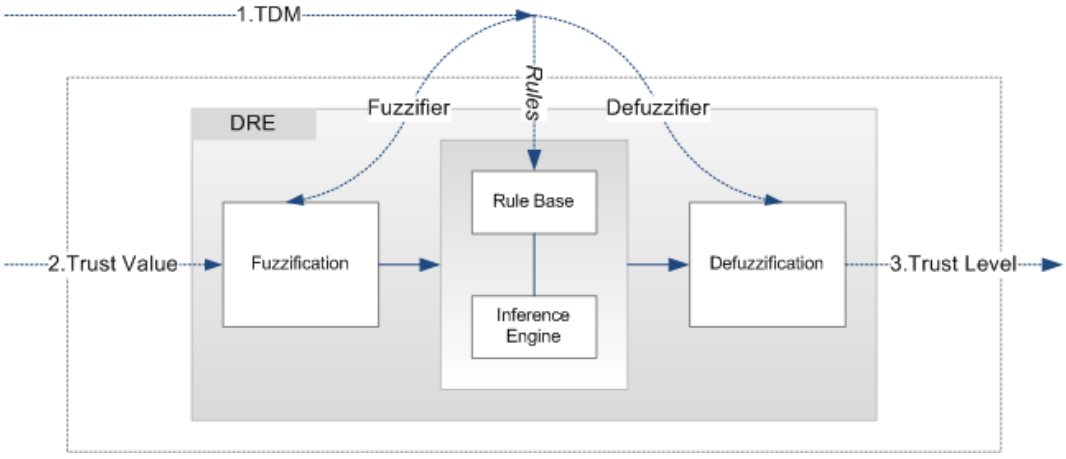


Figure 4.6: DRE Architecture

Fuzzification The input variable, trust value, is fuzzified using the *Fuzzifier* block defined in the DM. This requires a determination of the degree to which the input trust value $\overline{TV(R_i)}$ belongs to each of the predefined fuzzy sets via membership functions. The result of the fuzzification is a fuzzy *degree of membership* μ in the qualifying linguistic set x, $(\mu(x) \in [0,1])$. In the example illustrated in 4.7, the DM Fuzzifier block defines three *linguistic terms* comprising of a name as well as a membership function. For instance, the term *poor* uses a piece-wise linear membership function defined by points $(X_0 = 0.0, Y_0 = 1)$ and $(X_1 = 0.5, Y_1 = 0.0)$.

```
<Fuzzifier Name="trust_value">
  <Terms>
    <Term Name="poor">
    <Points>
      <Point X="0.0" Y="1.0" />
      <Point X="0.5" Y="0.0" />
    </Points>
    </Term>
    <Term Name="good">
    <Points>
      <Point X="0.0" Y="0.0" />
      <Point X="0.5" Y="1.0" />
      <Point X="1.0" Y="0.0" />
    </Points>
    </Term>
    <Term Name="excellent">
    <Points>
      <Point X="0.5" Y="0.0" />
      <Point X="1.0" Y="1.0" />
    </Points>
    </Term>
  </Terms>
</Fuzzifier>
```

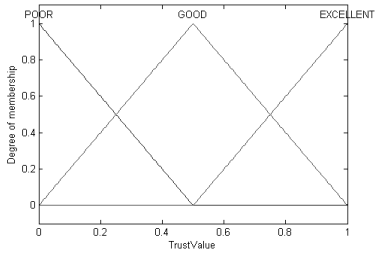


Figure 4.7: Fuzzification Membership Functions

Rule-Based Inference Engine The fuzzy inference process executes in three steps. In the first step, each of the rules in the *Rules* block defined in the DM is processed. Each rule defines a mapping between the fuzzification and the defuzzification blocks and uses an implication method, such as min (minimum) which truncates the output fuzzy set. In this example, three rules are defined, so the output of the first step is in a form of three truncated fuzzy sets.

Step 1: *Implication*

» **input**: Mapping rules + implication method » **output:** 3 truncated fuzzy sets

```
<TrustDecisionModel>
  ...
  <!-- tv: \gls{trust value} tl: \gls{trust level}
  <Rules>
    <Rule Id="1" Expression="IF tv IS poor THEN tl IS none" />
    <Rule Id="2" Expression="IF tv IS good THEN tl IS limited" />
    <Rule Id="3" Expression="IF tv IS excellent THEN tl IS full" />
  </Rules>
</TrustDecisionModel>
```

The subsequent step is to accumulate all output sets. accumulation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set using an accumulation method, such as max (maximum).

Step 2: *Accumulation*

» **input**: 3 truncated fuzzy sets + accumulation method » **output:** acc. fuzzy set

Defuzzification The final step is the defuzzification process. The input for the defuzzification process is the accumulated fuzzy set and the output is a single number. The DM defines a *Defuzzification method*, such as CoG (Centre of Gravity), which returns the centre of area under the curve to produce the trust level $\overline{TL(R_i)}$. The output variable *trust level* is defuzzified to get a crisp output number (between 0 and 1). This is defined using the *Defuzzifier* block of the DM. Similar to the fuzzification block, linguistic terms are defined. For instance, the term *none* uses a piece-wise linear membership function defined by points $(X_0 = 0.0, Y_0 = 1.0)$ and $(X_2 = 0.2, Y_2 = 0.0)$ illustrated in Figure 4.8. In addition, the Defuzzifier block requires a definition of an accumulation method and a defuzzification method.

```
<Defuzzifier Name="trust_level" AccumulationMethod="MAX"
    DefuzzificationMethod="COG" DefaultValue="0">
  <Terms>
    <Term Name="none">
    <Points>
      <Point X="0.0" Y="1.0" />
      <Point X="0.2" Y="0.0" />
    </Points>
    </Term>
    <Term Name="limited">
    <Points>
      <Point X="0.2" Y="0.0" />
      <Point X="0.5" Y="1.0" />
      <Point X="0.8" Y="0.0" />
    </Points>
    </Term>
    <Term Name="full">
    <Points>
      <Point X="0.8" Y="0.0" />
      <Point X="1.0" Y="1.0" />
    </Points>
    </Term>
  </Terms>
</Defuzzifier>
```

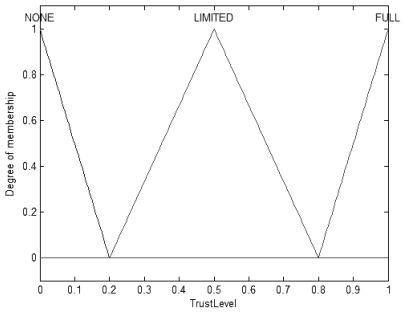


Figure 4.8: Trust Level Membership

Figure 4.9 illustrates the steps involved when mapping a trust value $\overline{TV(R_i)}$ onto the trust level $\overline{TL(R_i)}$ using the DM and DRE components (including 3 different samples). In addition, a surface graph is presented, demonstrating general trust value to trust level mappings based on the supplied DM.

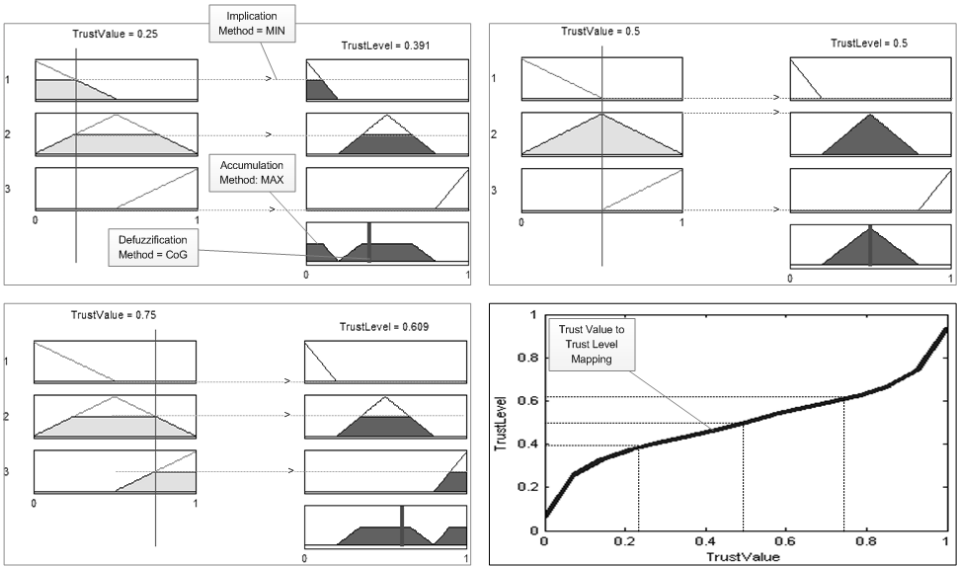


Figure 4.9: Mapping Trust Values to Trust Levels in MATLAB®

Reputation-Policy Report Generation

The final step involves generating the Reputation-Policy Report (RPR). During this step the RA iterates through the RVT and the Rule Degree Table (RuT) where for each resource, the resource identifier, the computed trust value and trust level are printed as well as the RuT values (Algorithm 1, lines 28-34). The reports is returned back to the calling client as an XML string (Algorithm 1, lines 37-38). The following listing is a sample RPR output of the querying mechanism:

```
<Report Timestamp="2009-10-26 20:06:07.734 GMT">
  <Resource Id="1" Value="0.42" Level="0.63">
    <Rules>
      <Rule Id="3" Degree="0.17"/>
      <Rule Id="2" Degree="0.0"/>
      <Rule Id="1" Degree="0.0"/>
    </Rules>
  </Resource>
</Report>
```

The RPR contains a reputation entry for a single Grid resource: R_1. The resource entry contains a trust value (0.42), computed by processing the evaluation model and trust level (0.63), computed by routing the trust value through the DRE and applying a set of decision rules. Each rule is printed with degree value indicating the membership of the trust level within that rule (e.g. 17% participation in rule 3 - 'EXCELLENT' and 0% participation in rules 1 and 2 - 'POOR' and 'GOOD').

4.3.3 VO Aggregated Reputation Querying Mechanism

Overview

The aggregated reputation querying mechanism provides a valuable extension to the Grid resource reputation querying mechanism. It allows trust aggregation on an organisational basis by obtaining the trust value for an organisation contributing one or more resources or even obtaining the reputation trust for an entire VO based on the individual trust of its members. In addition, it allows inferring the trust value for an individual resource based on the trust and reputation of the organisation the resource belongs to (i.e. resource inference). These properties are of great importance in the context of VO formation and operation phases as they assist in partnership formation and the binding of selected partners into the actual VO.

The basic principle behind the aggregated reputation querying mechanism is to leverage the aforementioned Grid resource reputation querying mechanism and apply it in a larger context. This serves the advantage of using the same TDS antecedently used for evaluating a single or a set of resources with the intention of evaluating a participating organisation or even an entire VO. Recall Formula 4.6, the overall trust value for an organisation Org is defined as the mean value of its contributed resources. This is denoted by Formula 4.9:

$$\overline{TV(Org_i)} = \frac{\sum_{i=1}^{|RVT|} \overline{TV(R_i)}}{\mid RVT \mid}, (1 \leq \mid RVT \mid, 1 \leq i \leq \mid Org \mid) \tag{4.9}$$

Where each resource in RVT is a resource contributed by organisation Org_i ($r \in RVT \rightarrow r \in Org_i$). Similarly, the overall trust value for a virtual organisation VO_i is defined as the mean value of its participating organisations $\{Org_1, \ldots, Org_n\}$:

$$\overline{TV(VO_i)} = \frac{\sum_{Org \in VO} \overline{TV(Org)}}{\mid Org \mid}, (1 \leq \mid Org \mid, Org \in VO_i) \tag{4.10}$$

Where the variables $\overline{TL(Org_i)}$ and $\overline{TL(VO_i)}$ refer to the aggregated trust level of an organisation and a VO correspondingly. In the context of this thesis, the aggregated trust value of an entity (e.g. entity, organisation, VO) is also referred to as *context value*. This stands in contrast to the term *trust value* (formula 4.6), which solely refers to the trust value of a single entity.

In order to simplify the annotation used throughout the remaining sections, the variable $\overline{TV(C_c)}$ shall be used to define the trust value of an arbitrary context C_c ($1 \leq c \leq \infty$). correspondingly, $\overline{TL(C_c)}$ refers to the trust level of context C_c.

GREPTrustAggregator

Overview Considering a VO comprising of M underlying organisations where each organisation contains N resources, the computational complexity of processing the evaluation model for the VO is $O(M \times N)$. In order to reduce the execution time of such computation, the GREPTrustAggregator (GTA) component is introduced. The rationale behind incorporating GTA is to utilise and control multiple instances of GREPTrust, where each instance computes in parallel a portion of the evaluation model input dataset (i.e. a subset of the resources or a subset of the participating organisations) thus scaling the GREPTrust solution for very large grids.

Solution Figure 4.10 illustrates the GTA solution. In contrast to Figure 4.1, in this constellation the resource broker contacts the GTA component instead of the GREPTrust component. This also simplifies the reputation evaluation process from the resource broker perspective - instead of contacting and aggregating the results from multiple GREPTrust instances, the GTA acts as a façade to the aggregation solution, thus hiding the complexity involved from the resource broker. Figure 8 in appendix .2.1 illustrates a UML class diagram of GREPTrustAggregator.

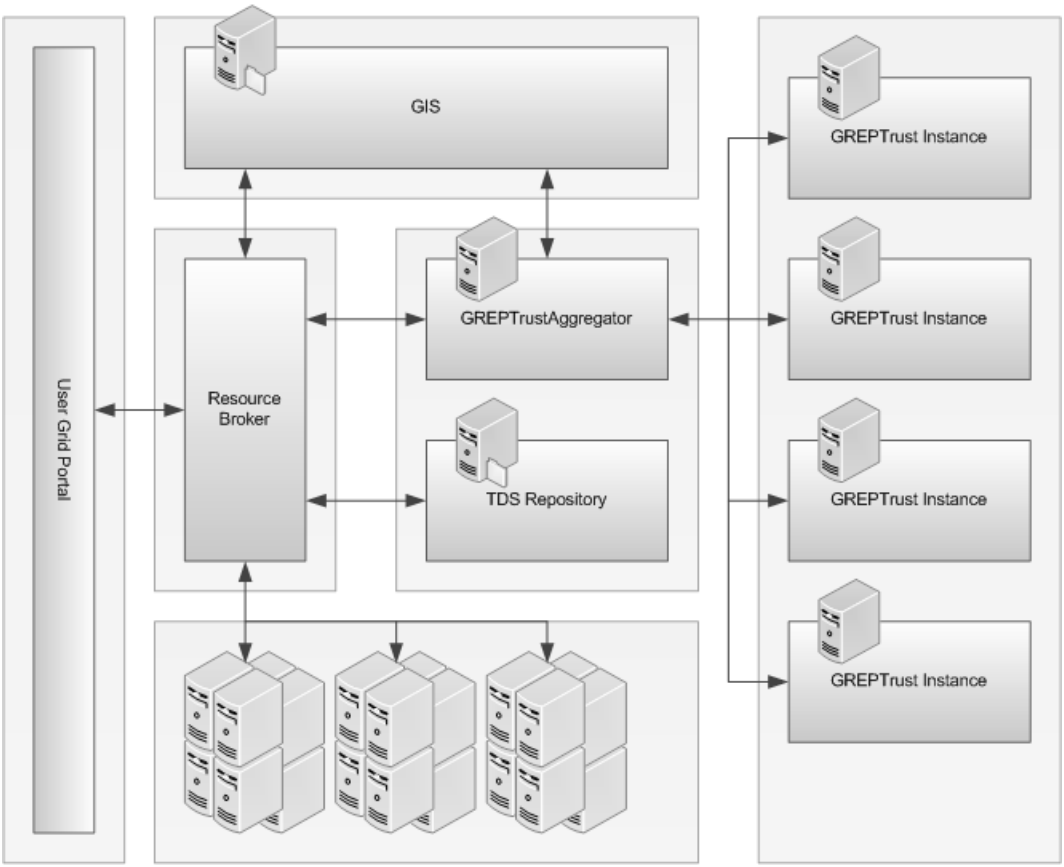


Figure 4.10: GREPTrustAggregator Solution

System Components Figure 4.11 illustrates the GTA architecture. It retains resemblance to the QM architecture (Figure 4.4) as it also comprised of three steps for processing an RPQ: (i) Process TDS Evaluation Model (Algorithm 6, lines 6-9), (ii) Process TDS Decision Model (Algorithm 6, lines 10-11) and (iii) Generate Reputation-Policy Report (Algorithm 6, lines 12-13). As a matter of fact, steps (ii) and - (iii) are identical between the GTA and QM architectures. Therefore, the remaining sections will concentrate on GTA step (i), comprising of GTAggregator, ResourceAggregator, GTProxy and GTResourceAllocator.

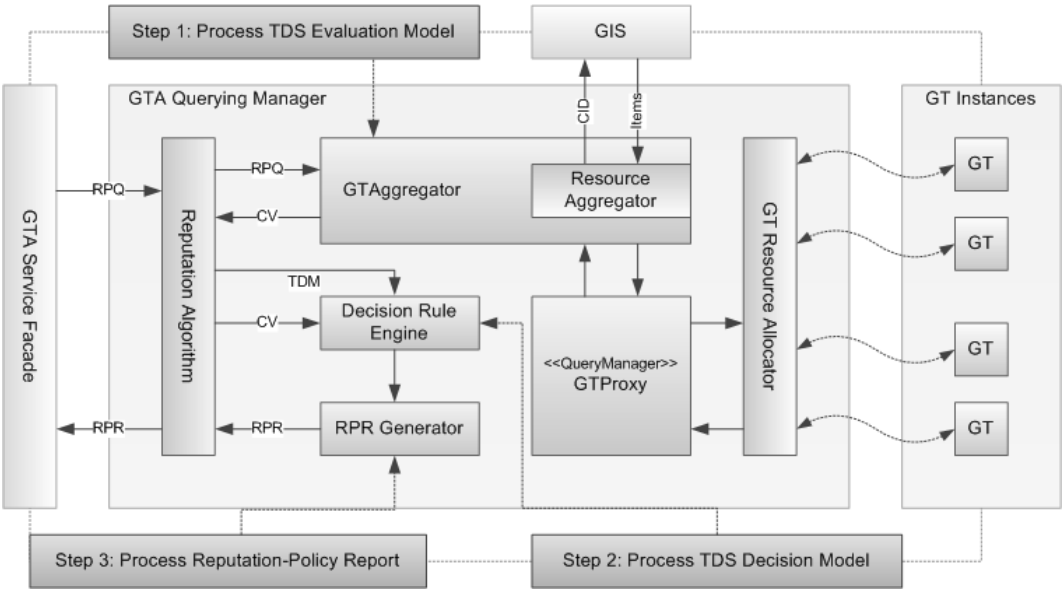


Figure 4.11: GREPTrustAggregator Architecture

Algorithm 6: RA::Query(RPQ) algorithm

1: RPQ: reputation-policy query
2: DM: trust decision model
3: CV: context value
4: Rep: report entity
5: RPR: reputation-policy report
6: $CV \leftarrow ProcessEvaluationModel(RPQ)$
7: **if** $CV = -1$ **then**
8: **return** *null*
9: **end if**
10: $DM \leftarrow DM(TDS(RPQ))$
11: $Rep \leftarrow ProcessDecisionModel(CV, DM)$
12: $RPR \leftarrow RPRGenerator(Rep)$
13: **return** *RPR*

TDS Evaluation Model Processing The TDS evaluation model processing step is primarily managed by the GTAggregator component. The GTAggregator component accepts an RPQ as an input parameter and returns a context value - *CV*. In the aggregative model, the resource identifier field in the RPQ stores the evaluated context ID - *CID* (such as organisation or VO). The advantage of this approach is that it retains the RPQ structure without a need to introduce a specific field for context ID. Algorithm 7 describes the top level steps required for processing the evaluation model. It is comprised of the following functions: (i) ResourceAggregator() - (Algorithm 7, line 6), (ii) BuildDepthMeanValues() - (Algorithm 7, line 10) and (iii) ProcessDepthMeanValues() - (Algorithm 7, line 11).

Algorithm 7: GTA::ProcessEvaluationModel(RPQ) algorithm

1: RPQ: reputation-policy query
2: CID: context ID
3: CT: context table
4: TM: total means
5: $\overline{TV(C_c)}$: trust value for context c
6: $CT \leftarrow ResourceAggregator(CID)$
7: **if** $Size(Keys(CT)) = 0$ **then**
8: **return** -1
9: **end if**
10: $BuildDepthMeanValues(RPQ, items)$
11: $TM \leftarrow ProcessDepthMeanValues()$
12: $\overline{TV(C_c)} \leftarrow MeanValue(TM)$
13: **return** $\overline{TV(C_c)}$

ResourceAggregator The purpose of the ResourceAggregator component is to translate given context ID - *CID* into a tree data structure where each node contains an underlying resource ID. In order to achieve this structure, it is assumed that the ResourceAggregator utilises a Grid Information System (GIS) such as MDS [89] or R-GMA [88, 90] (appendix .2.2). This is converted into a context table *CT*:

```
CT = {
  D2 -> {
    104=[17, 18, 19, 20, 21],
    103=[14, 15, 16],
    102=[10, 11, 12, 13],
    101=[3, 4, 5, 6, 7, 8, 9]}
  D1 -> {
    100=[1, 2, 22]}}
```

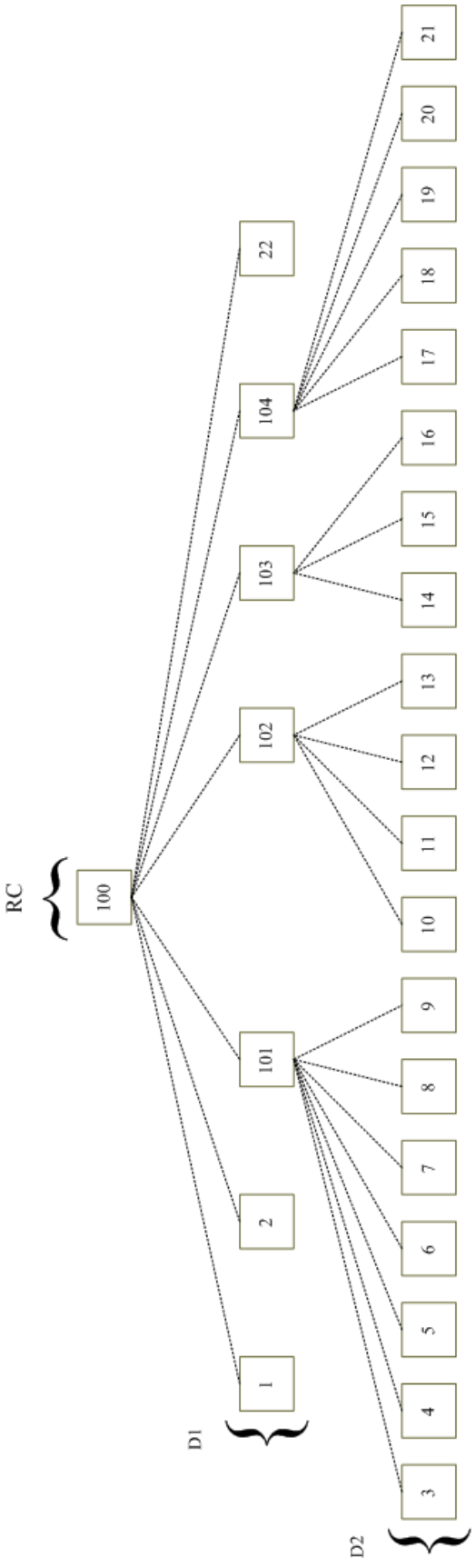


Figure 4.12: Sample context hierarchy produced by a GIS

Based on Figure 4.12, the context table *CT* is organises the underlying resources by the depth level of the tree. It is essentially a hash table where the keys are the depth level and the values is an additional hash table comprising of resource root context *RC* and its contained elements. This is achieved using a post-order traversal.

BuildDepthMeanValues The purpose of the BuildDepthMeanValues function is to iterate through the context table *CT* and produce a Depth Mean Value (*DMV*). The DMV is a hash table comprising of the depth level of the tree as the key and an array of average values of all siblings within the same depth level as the value. In the following listing example, in depth 2, 0.44 is the mean value of of resources 3-9, 0.59 of 10-13, 0.61 of 14-16 and 0.46 of 17-21. In depth 1, 0.46 is the mean value of resources 1, 2 and 22. Algorithm 8 describes the BuildDepthMeanValues function.

```
1 DMV={ 2=[0.44, 0.59, 0.61, 0.46], 1=[0.46] }
```

Algorithm 8: GTA::BuildDepthMeanValues(RPQ, items) algorithm

```
1: CT: context table
2: DMV: depth mean value
3: RC: root context
4: KVP: key value pair
5: DIndices ← ReverseKeys(CT)
6: for index = 0 to Size(DIndices) do
7:    Depth ← DIndices[index]
8:    RC ← Items.Get(depth)
9:    for all Res ∈ KeySet(RC do
10:      Resources ← RootCtx.Get(resource)
11:      SetResources(RPQ, Resources)
12:      RPR ← Invoke(RPQ)
13:      Mean ← MeanValue(RPR)
14:      KVP ← GetDepthKeyValues(Depth, Mean)
15:      Put(DMV, KVP)
16:   end for
17: end for
```

In order to obtain the trust value for each resource set in each depth level, the *Invoke* function is used on available GREPTrust instances (Algorithm 8, line 13). The returned PRR is extracted and a mean value function is performed on it (Algorithm 8, line 14) and finally inserted into the DMV hash table (Algorithm 8, line 16). The GREPTrust invocation function is further described in the following section.

GREPTrust Invocation The purpose of the GTA invoke method is to allocate an available GREPTrust resource and utilise it for querying a depth entry (e.g. $103 \mapsto \{14,15,16\}$). In order to submit a query a GREPTrust instance, the GTResourceAllocator component pools a collection of predefined GREPTrust instances and allocates an idle instance (Algorithm 9, line 3). A reference to the allocated GREPTrust instance is stored inside the GTProxy component, forming an interface to the actual GREPTrust object (i.e. proxy design pattern). The GTProxy is invoked with the modified RPQ - i.e. containing a subset of resources (Algorithm 9, line 5) and eventually returns an RPR (Algorithm 9, line 7).

Algorithm 9: GTA::Invoke(RPQ) algorithm

```
1: RPR: reputation-policy report
2: RPR ← null
3: Allocator ← Instance(GTResourceAllocator)
4: GTProxy ← Allocate(Allocator)
5: if GTProxy ≠ null then
6:    RPR ← GTProxy(RPQ)
7: end if
8: return RPR
```

ProcessDepthMeanValues The purpose of the ProcessDepthMeanValues function is to iteratively calculate the mean trust value at each depth level $(n, n-1, \ldots, 1)$ and produce an array of the total mean values *TM* at depth level 1.

Algorithm 10: GTA::ProcessDepthMeanValues() algorithm

```
1: TM: total means
2: if Size(DMV) > 0 then
3:    DMVKeys ← ReverseKeys(DMV)
4:    for index = 0 to Size(DMVKeys) do
5:       MeanDepth ← DMVKeys[index]
6:       Values ← Get(DMV, MeanDepth)
7:       TotalMean ← MeanValue(Values)
8:       Add(TM, TotalMean)
9:    end for
10: end if
11: return TM
```

```
1 TM=[0.53, 0.46]
```

Once the *TM* is produced, a mean value function is performed on its elements (Algorithm 7, line 12) which produces the final trust value for the given context $\overline{TV(C_c)}$. Using the listing above, this value would be $\frac{(0.53+0.46)}{2} \cong 0.5$

Grid Resource inference

Grid resource inference allows inferring about the trustworthiness of a single resource based on the global context, such as an organisation the resource belongs to or the VO in which it participates. This is useful in scenarios where the trust level of an arbitrary resource has to consider the global context, which is used as an index measure. Formula 4.11 defines the trust level for a context-aware resource $\overline{TL(R_i)_{CA}}$:

$$\overline{TL(R_i)_{CA}} = \Phi(\overline{TV(R_i)}, \overline{TV(C_c)}), (R_i \in C_c) \tag{4.11}$$

It retains similarity to the standard trust level formula, (formula 4.8). However, in formula 4.11, the fuzzy inference function Φ, accepts an additional parameter $\overline{TV(C_c)}$, which refers to the trust value of context C_c of which resource R_i belongs to. The trust level of R_i is marked with CA in order to emphasise that the trust level is context-aware. Figure 4.13 illustrates a constellation for Grid resource inference. In this type of settings, two reputation-policy queries are performed in a series, where the output of the first query (containing the aggregated context value - CV), serves as an input for the second query. The required steps are performed as follows:

1. The resource broker obtains $TDS1$ from the TDS repository. $TDS1$ contains reputation evaluation criteria for an aggregated context.

2. The resource broker constructs a reputation-policy query - $RPQ1$ comprising of $TDS1$ and submits it to an instance of GREPTrustAggregator.

3. GREPTrustAggregator processes the aggregated query and returns a reputation-policy report - $RPR1$ for the aggregated context CV $(0 \leq CV \leq 1)$.

4. The resource broker obtains $TDS2$ from the TDS repository. The DM section of $TDS2$ contains two *Fuzzifier* subsections - one for the fuzzification of the produced trust value of the evaluated resource and the second - for the fuzzification the context value. This approach evaluates the trust level whilst taking into account the global context (a sample TDS is available in appendix .2.2).

5. The resource broker constructs a new reputation-policy query $RPQ2$ which contains $TDS2$ and CV. The $RPQ2$ is submitted to an instance of GREPTrust.

6. GREPTrust returns the a reputation policy report $RPR2$, which contains context-adjusted trust level TL for the evaluated resource $(0 \leq TL \leq 1)$.

The advantage of this constellation is that it utilises and interlinks GREPTrust and GREPTrustAggregator components without any need to introduce a specific component for Grid resource inference. The only two modifications required are in

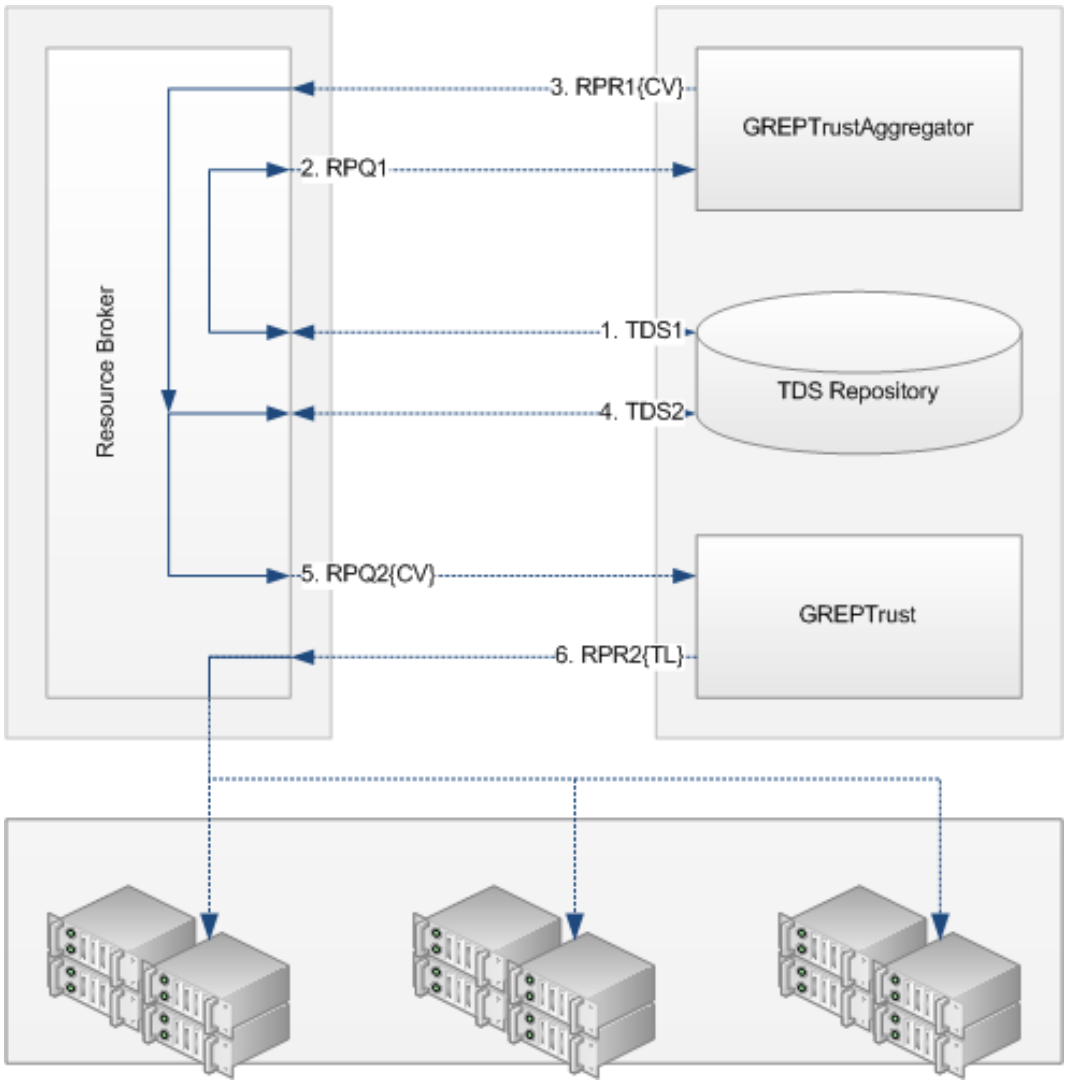


Figure 4.13: Grid Resource Inference Solution

the DM aspect of the TDS: (i) introduction of context value membership functions in the *Fuzzifier* section and (ii) additional IF-THEN rules in the *Rules* section.

Figure 4.14 illustrates a surface graph demonstrating the mappings between trust and context value to trust level. An interesting feature of the graph is that even if the trust value is relatively high (> 0.5), the trust level still remains low as long as the context value is low (< 0.5). In contrast, if the context value is very high (> 0.7) and the trust value is very low (approaching 0), the overall trust level remains fairly high (between 0.5 and 0.6). This shows how adjustments are made in boundary cases where trust value rankings are extremely above or below the context value.

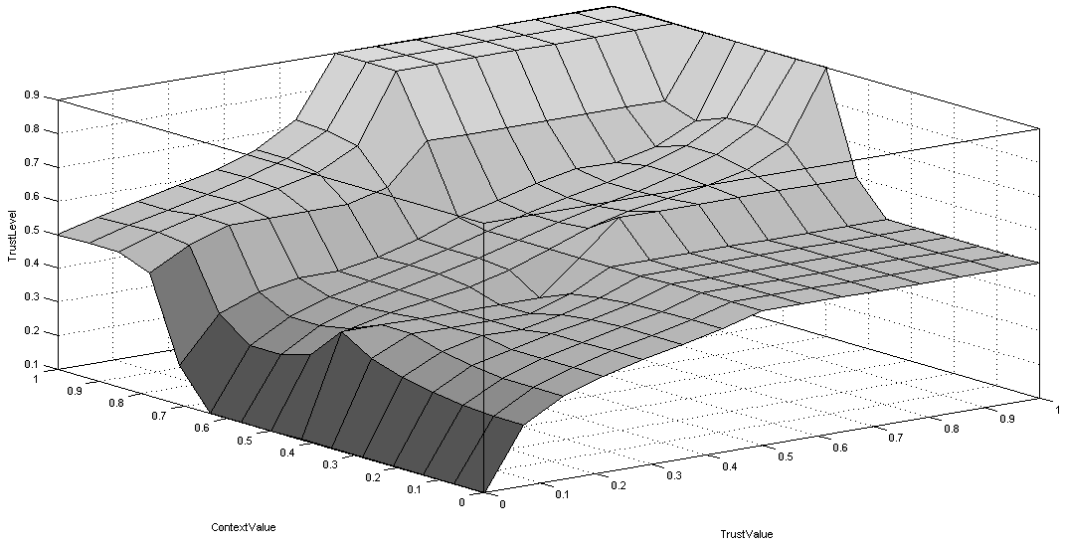


Figure 4.14: Mapping Trust & Context Values to Trust Levels in MATLAB®

4.4 Conclusions

The main objective of this chapter was to introduce the Grid Reputation-Policy Based Trust Management Service (GREPTrust), including its functional requirements within the context of virtual organisations and system components. An additional aim of this chapter was to introduce the Grid reputation-policy based querying mechanism, which constitutes as a central aspect of GREPTrust. This included an analysis of the process of evaluating the trustworthiness of a single or a set of resources as well as an analysis regarding the process of performing aggregated reputation queries. In conclusion, the main outputs of this chapter are the following:

- Elaborative description of the architectural components of GREPTrust. This constitutes as the second research contribution: *RC2: Grid reputation-policy trust management service architecture.*
- Process analysis of the Grid resource reputation querying mechanism. This constitutes as the third research contribution: *RC3: Grid resource reputation querying mechanism.*
- Process analysis of the VO Aggregated reputation querying mechanism including Grid resource inference. This constitutes as the forth (and final) research contribution: *RC4: VO aggregated reputation querying mechanism.*

Chapter 5

GREPTrust Testbed, Case Study & Simulation Design

"The distance between insanity and genius is measured only by success."
Bruce Feirstein

5.1 Introduction

This chapter serves three purposes: (i) introduce the GREPTrust testbed architecture by discussing the design requirements for allowing side-by-side comparison of GREPTrust exoteric reputation model with the GridPP esoteric reputation model; (ii) present a case study which is highly applicable to the financial industry and utilises the GREPTrust testbed; (iii) describe a workflow simulation based on the business requirements and comparison strategies identified by the case study. The simulation is conducted by executing two Grid jobs; the first job utilises the resources recommended by GridPP while the second job - by GREPTrust.

Therefore, this chapter is divided into three sections - GREPTrust Testbed, Case Study and Simulation Design. The first section describes the testbed environment including system constraints and resolutions as well as the mandatory steps required for producing a Trust Comparison Report (TCR) listing the resources evaluated by GridPP and GREPTrust. The second section describes a real-world computational finance case study, which includes a problem domain, scenario description and comparison strategy. The third section describes the steps required for the workflow simulation, including symbol data generation, VDP jobs execution and comparison analysis. This section concludes with a representation of the simulation results.

5.2 GREPTrust Testbed

5.2.1 Overview

The implementation 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 an efficient deployment of the synergistic reputation-policy based trust model as a Grid service and simulate interactions between Grid clients, GREPTrust and Grid resources. The objective of this testbed is to constitute as an automated testing framework for comparing Grid reputation models as well as for revealing if the reputation-policy paradigm assists in improving resource selection under mission critical scenarios. The initiative behind this trust evaluation exercise is to vigilantly observe the testbed procedures required for producing a *Trust Comparison Report* (TCR), in which every evaluated Grid resource is associated with the trust levels assigned to it by both GREPTrust and GridPP reputation models. Consequently, the testbed execution workflow is comprised of three consecutive steps: feedback data generation, TDS selection and Resource evaluation.

Feedback Data Generation The feedback data generation step is concerned with creating historical ratings feedback values for each of the supported opinions (i.e. availability and reliability). This step is of paramount importance as both reputation models solely rely on feedback data in order to calculate trust levels. Even though obtaining production feedback data from the UK SAM test results would be highly captivating, it would obstruct the versatility and flexibility in generating scenario driven data (e.g. high reliability/low availability), resulting in limited number of combinatorial test cases. Essentially, this step is a central part of a larger scope data generation process which involves creating values for the entire data model, including opinions, clients, resources, executions as well as feedbacks. Therefore, the total number of feedbacks is denoted by the following formula:

$$\begin{cases} N_{executions} = N_{clients} \times N_{resources} \times N_{records} \\ N_{feedbacks} = N_{executions} \times N_{opinions} \end{cases} \tag{5.1}$$

The number of executions is the product of the number of clients, the number of resources and the number daily recorded executions. Consecutively, the number of feedbacks is the product of the number of executions and the number of opinions. For the testbed simulation purposes: 2 opinions (availability and reliability), 10 clients, 22 resources (equal to the number of GridPP participating sites) and 681 daily executions were defined. Therefore, the number of executions is 10 x 22 x 681 = 149,820 records and the number of feedbacks is 149,820 x 2 = 299,640 records.

The feedback data samples are drawn upon an approximated normal (Gaussian) distribution for each of the 22 participating resources. This function is supplied with 3 parameters: The random variable x representing the number of resources (i.e. 22), the mean μ controlling the centre point of the distribution and variance σ controlling the spread of the feedback values within the distribution:

$$f(x;\, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{(2\sigma^2)}} \tag{5.2}$$

Essentially, x represents the number of the participating sites in the VO for which to generate feedback samples for (i.e. 22 feedbacks), μ represents the VO average feedback value and σ represents the volatility of the VO. Since the permissible range of the feedback distribution must conform to the rule: $f(x;\, \mu, \sigma^2) \in [0,1]$, the function is normalised in the following way:

$$P_{vo}(x) = \begin{cases} 0 & \text{for } f(x;\, \mu, \sigma^2) < 0; \\ 1 & \text{for } f(x;\, \mu, \sigma^2) > 1; \\ f(x;\, \mu, \sigma^2) & \text{otherwise;} \end{cases} \tag{5.3}$$

so that every feedback value greater than 1 is normalised to 1 and every feedback value less than 0 is normalised to 0. The value $P_{vo}(x)_{[i]}$ represents the average feedback value for resource i. This feedback value serves as the mean parameter (μ) for a similar distribution function which generates the feedback distribution made for that particular resource (repeated for availability & reliability). It is calculated in the following way:

$$P_r(x) = \begin{cases} x = 6810; & \text{(10 clients x 681 daily executions)} \\ \mu = P_{vo}(x)_{[i]}; & \\ \sigma = 0.02; & \text{(constant value)} \\ f(x;\, \mu, \sigma^2) & \end{cases} \tag{5.4}$$

Figure 5.1 illustrates a scatter chart which compares the average availability and reliability values for each of the 22 simulated testbed resources. As can be seen from the diagram, the overall reliability value is significantly higher ($\mu = 0.9$) as well as significantly more stable ($\sigma = 0.01$) than the overall availability value (useful for simulating non-available yet reliable resources).

TDS Selection The TDS selection step is concerned with manually fine tuning the trust evaluation and decision criteria to be submitted along to the heuristic

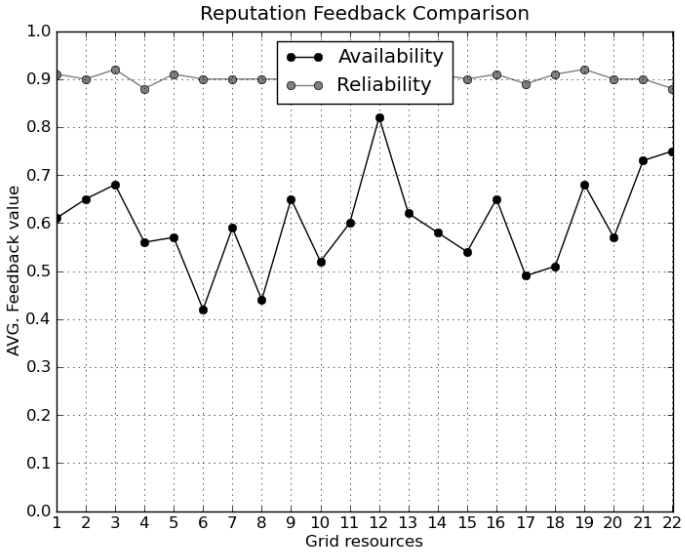


Figure 5.1: Feedback comparison example

GREPTrustResource instance. As the deterministic instance is provided with a constant TDS in each execution, the heuristic GREPTrustResource instance, on the other hand, can be supplied with an optimised TDS considering different constraints, such as job requirements, opinion feedback data (average value & standard deviation), selection threshold, execution time (translated to minimum number/ratio of resources) and etc. For example, considering a Grid resource with the following attributes: avg. availability value = 0.3; avg. reliability value = 0.9. Suppose the evaluation model initially dictated equal weight on availability and reliability (i.e. 50%) due to the job requirements, then the aggregated trust value would be 0.6. However, assuming that the constraints: threshold value = 0.7; min. resource ratio = 100% implying that the decision model can be used in order to map trust value of 0.6 (and above) to trust level 0.7 in order to meet the requirements.

Resource Evaluation The resource evaluation step is concerned with processing both TDS files and generating a trust evaluation comparison. In Figure 5.2, both heuristic and deterministic GREPTrustResource instances are being compared. The horizontal axis lists Grid resources 1 to 22 and the vertical axis denotes the trust level. The circle and square series points represent GridPP and GREPTrust trust levels 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 0.6 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 their counterparts evaluated by GridPP. Out of 22 resources, only 2 resources were reported competent enough for selection by GREPTrust, whilst 10 resources were reported competent by GridPP. This demonstrates how GREPTrust can reduce the risk of job execution failure using external reputation evaluation criteria.

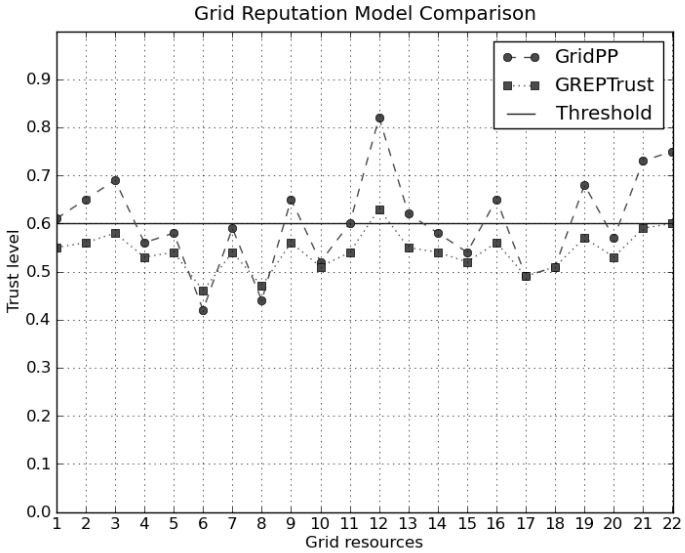


Figure 5.2: Model comparison sample

5.2.2 Infrastructure

The GREPTrust testbed architecture was designed for comparing the GREPTrust reputation model with an existing reputation model for proving the advantages of stipulating reputation requirements when qualifying resources for selection. The comparison to an existing model is challenging as GREPTrust 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 address this challenge, it has been decided to deploy two instances of GREPTrust. The first instance represents a heuristic approach where it is capable of accepting a different TDS file with each execution depending on the historical feedback data, VO characteristics, the job requirements and risk tolerance - all acquired prior to any execution. The second GREPTrust instance represents a deterministic approach, which is modelled after the reputation algorithm used by the GridPP project. This instance uses constant TDS configuration consisting of

permutation of a single opinion (availability), equal weight rules and no decision model, effectively mapping trust values as trust level values. The reason for this configuration is justified by the nature of the GridPP reputation algorithm.

While being used by the UK SAM tests for maintaining availability and reliability historical feedback data regarding the participating sites, the GridPP reputation algorithm executes test jobs on regular basis in order to continually assert the minimum functionality required for a site to participate in the computation. Each batch of tests measures the Computing Element (CE), Storage Element (SE) and Storage Resource Manager (SRM) of each participating site and records the aggregated feedback value reflecting its overall performance. In GridPP, when a Grid client such as a resource broker or a job scheduler acquires resources for job submission it is capable of contacting the reputation algorithm and obtain two feedback datasets (availability and reliability factors) for each of the participating sites. However, only one dataset can be used at any time. In other words, since a consolidation option is not supported by the GridPP reputation algorithm, only one the of the reputation factors is selected (i.e. availability) for evaluating Grid resources and in addition, options to stipulate weight rules and Decision Model (DM) *trustvalue* $\rightarrow$ *trustlevel* mappings are omitted in order to model the crude state of the feedback data.

Figure 5.3 illustrates the GREPTrust testbed architecture. It is comprised of six logical packages: GREPTrustAnalysis, GREPTrustResource, GREPTrustAPI, GREPTrustTestBed, GREPTrustAdmin and GREPTrustDB. The central package is GREPTrustTestBed, which is implemented on top of the GridSim simulation environment [76]. Since GridSim allows streamlined deployment of software components as Grid resources, it was found highly suitable for providing infrastructure services for GREPTrustTestBed. Consequently, GREPTrustTestBed deploys 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. GREPTrustDB data model is managed by GREPTrustAdmin, which contains the relevant logic for generating the ratings feedback data structure as well as generating and populating that data. GREPTrustTestBed contains the simulation logic used for initialising the simulation environment, instantiating the two GREPTrust resources and a generic Grid client constructing two reputation-policy queries, submitting each query to a GREPTrust resource and storing 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 for preparing each scenario simulation, controlling the GREPTrust testbed using the JPype [132] bridge library and plotting simulation results to MATLAB® for further data analysis.

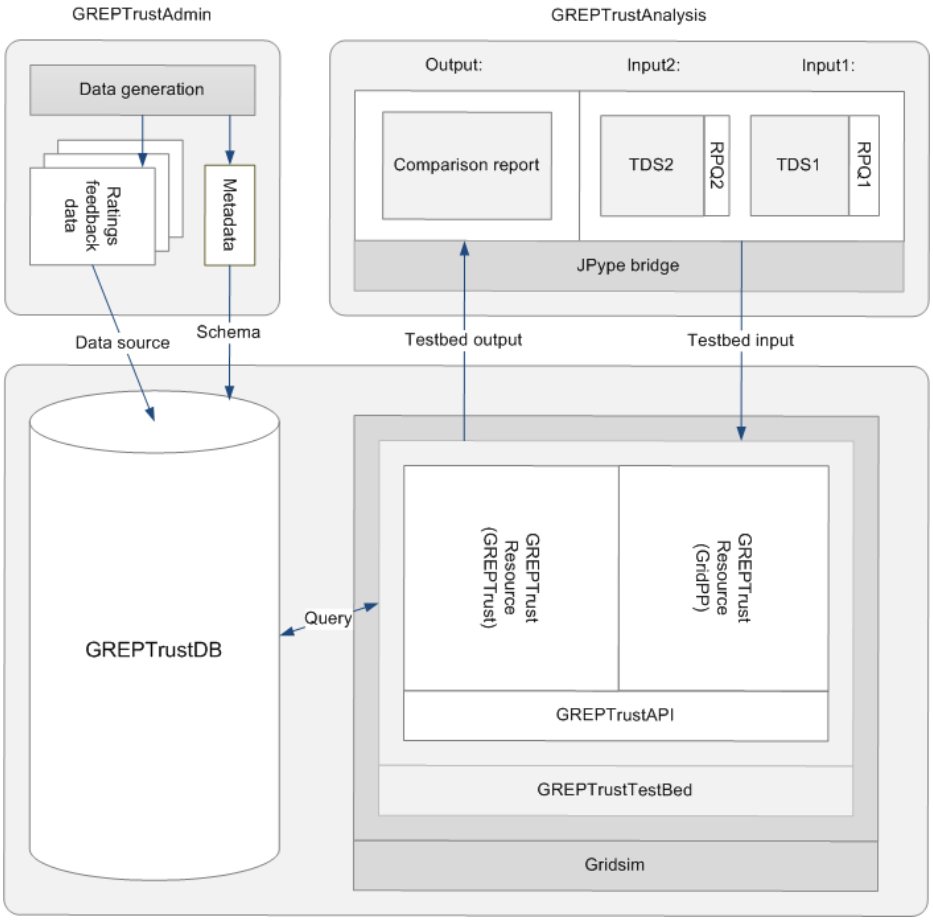


Figure 5.3: GREPTrust testbed architecture

Table 5.1 lists GREPTrust testbed packages and their architecture component counterparts. The GREPTrust testbed packages are roughly correspondent to the service architecture subsystems (client, system and data) as they provide concrete implementation for the architecture specification yet with additional functionality considering the testbed. For example, GREPTrustAdmin auxiliary methods for generating random sample data used as ratings feedback in addition to the data querying logic described in the architecture specification. In contrast, the GREPTrustTestBed package does not directly map to any component since it is a container package for both GREPTrustResource instances as well as GREPTrustAPI. Appendix .3.1 contains a diagram demonstrating the dependencies between the packages.

5.2.3 System Constraints & Resolutions

Since essentially GREPTrust is both an infrastructure service and an intermediate process within a supply chain (resource selection matchmaking process), it is subjected to several design and implementation constraints. Table 5.2 summarises

Package/component mapping		
Package	**Sub.Sys.**	**Component**
GREPTrustAnalysis	Client	Grid client & TDS repository
GREPTrustResource	Service	Reputation-Policy Service Façade
GREPTrustAPI	Service	Querying Manager (QM)
GREPTrustTestBed	Service	n.a. - no matching component in GREPTrust
GREPTrustAdmin	Service	Admin Manager (AM)
GREPTrustDB	Data	Reputation-Policy Data Store (RPDS)

Table 5.1: GREPTrust testbed architecture mapping

the testbed constraints and their resolutions & assumptions measurements which are taken in order to eliminate signs of vagueness regarding the testbed design.

#	**Constraint**	**Resolution**
C1	Deploying GREPTrust as a Grid service.	Decision was made to design the GREPTrust testbed architecture which leveraged the GridSim simulation environments as an underlying infrastructure.
C2	Comparing GREPTrust heuristic model to GridPP deterministic model.	Decision was made to deploy two instances of GREPTrust. The first instance (heuristic) uses varying TDS based on different constraints while the second (deterministic) uses constant TDS with each execution.
C3	Simulating feedback data generation for each of the supported opinions (availability & reliability).	Feedback sample data drawing using an approximated normal (Gaussian) distribution for each of the participating resources.
C4	Deciding on an optimised TDS given client/job criteria.	Assumption is given so that the selection of an optimised TDS reflecting the client/job criteria is performed prior to the reputation query (i.e. TDS optimisation is outside the scope of the research).
C5	Deciding on the final set of resources.	Assumption is given so that the final set of resources is subjected to additional set of constraints which may be imposed due to hardware/software requirements, regional & policy settings and etc.

Table 5.2: GREPTrust testbed constraints & resolutions

5.3 Case Study

5.3.1 Overview

The purpose of this case study is to provide a meaningful, tangible context for evaluating the reputation-policy trust model. The problem domain for this case study is taken from an increasingly popular field in computational finance known as algorithmic trading [101]. This term refers to the utilisation 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. 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.

These algorithms encapsulate complex quantitative models which rely on historical analytics for determining the algorithm behaviour. The integrity and validity of the data supplied by these analytics is of paramount importance for the performance of the trading algorithms. Corrupt values are generally hard to detect during pre-trade hours and may result in unexpected trading behaviours and financial loss.

The presented case study is focused on generating an historical volume distribution profile (VDP) for a VWAP trading algorithm [102]. The process for generating this type of analytics typically consists of segmenting each stock trading day into time intervals (bins) and computing the average percentage of daily volume traded during each bin. Figure 5.4 illustrates an example of a volume distribution profile for Vodafone Group Plc. The horizontal axis denotes the bin series whilst the vertical axis denotes the percent of volume. In this example, the trading day is segmented into 36 fifteen minute bins and each bar represents a 21 day average percent of daily volume traded during that bin. Being a scheduled driven trading strategy, the VWAP trading algorithm processes orders over specified time horizon and spreads each trade in proportion to the generated historical volume distribution. Extreme sensitivity to accurate generation of volume profiles and impact on strategy performance make the historical volume distribution profiles analytics suitable subject for assessing and evaluating the performance of competing reputation models.

5.3.2 Scenario Description

An international agency brokerage firm offers algorithmic execution services for its clients, which typically consist of large investment funds and asset managers. The algorithmic trading suite automates trading executions based on investment strategy and desired asset class. The entire suite relies on a daily historical volume distribution profile for processing trading executions for all shares (i.e. symbols).

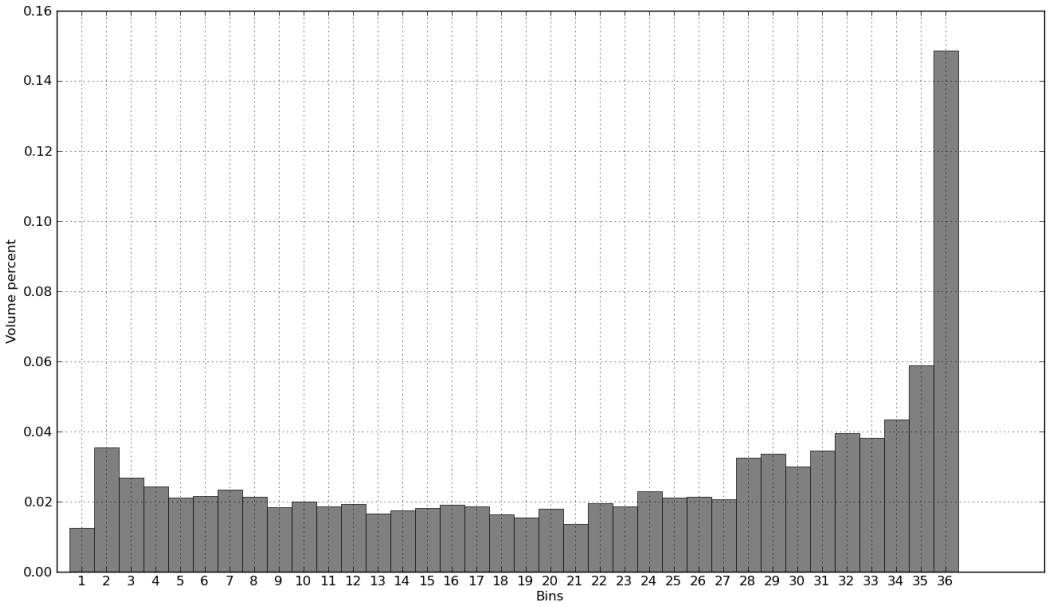


Figure 5.4: Volume distribution profile for Vodafone Group Plc

An overnight process handles the generation of historical volume distribution profiles for each symbol within the symbol universe (6,466 names in total). The process outputs a single volume distribution profile comprising of 232,776 entries ($6{,}466 \times 36$ bins) where each entry contains a compound key of a SEDOL (symbol unique identifier) and bin number in the left column (e.g. 4119054.bin1) and a value depicting the 21 day average percent of traded volume for the given SEDOL and bin number (about 3% of total volume for 4119054.bin1), as listed in Table 5.3.

SEDOL.bin#	**% of Traded Volume**
4119054.bin1	0.031100
4152941.bin1	0.000000
4295374.bin1	0.019900
4356033.bin1	0.095100
4359690.bin1	0.013600
...	...

Table 5.3: Sample volume distribution profile content

Once generated, the volume distribution profile is then multi-casted across the network into each Algorithmic Trading (AT) server approximately one hour before the market opening time (08:00 am). Due to the computational intensive nature of this process, it has been decided to utilise an enterprise Grid infrastructure which would split the profile generation process across multiple nodes and result in higher throughput accordingly. Figure 5.5 illustrates a high-level view of underlying Grid infrastructure components. SE_1 is a storage element which contains the symbol

universe. CE_1-CE_N are computing elements (CE) which represent parallel running computing nodes, each of which processes a portion of the symbol universe. SE_2 is a storage element (SE) which contains the assembled volume distribution profile.

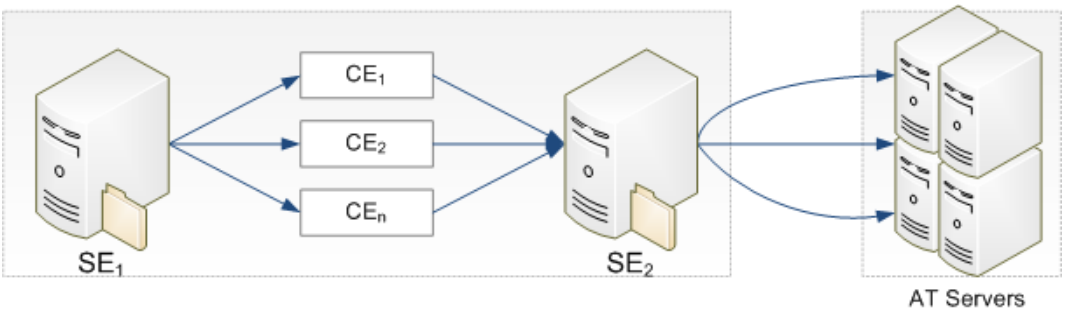


Figure 5.5: Proposed Grid infrastructure for improved VDP generation

During user acceptance testing (UAT), it has been discovered that although the usage of an enterprise Grid infrastructure has decreased the overall time required for generating the volume distribution profile, frequent resource computation and network errors resulted in either missing or corrupt entries in the file, consecutively leading to anomalies in the trading algorithms behaviour such as over/under/non trading during different fragments of the trading day. As an immediate step, it has been decided to implement a failover mechanism which switched to the previous day profile in case the output of the generation process contained missing entries. Despite being a suboptimal solution (as bin values tend to change on a daily basis), it assisted when one or more CE nodes were not available. A greater challenge required addressing erroneous entries due to reliability issues within the CE nodes, as it is virtually impossible to determine whether a bin value is corrupt or not.

This situation has raised a business demand for investing in a reputation-based trust management system which is efficient in filtering out low quality computing resources as well as flexible enough to cater for specific job requirements (as the proposed enterprise Grid infrastructure may be utilised for generating different types of analytics). Intensive business analysis has produced the reputation requirements for generating historical volume distribution profiles (summarised in Table 5.4). Since availability is already addressed by the failover mechanism, it is set to low priority. Job running time is rather flexible (as long as it completes before the market opening time). This provides room to select the highest quality resources. In contrast, reliability is a critical factor and is therefore set to high priority.

Quality Factor	Priority
Availability	Low
Job running time	Medium-Low
Reliability	High

Table 5.4: Quality factors and priorities

5.3.3 Comparison Strategy

The consecutive task was to compare competing reputation models based on the requirements specified in Table 5.4. Since GREPTrust allows external involvement in the trust and reputation evaluation process, a specialised TDS had been prepared in order to cater for the specific job requirements:

```
1  TrustDecisionStrategy {
2    EvaluationModel {
3      Availability := 0.1;
4      Reliability := 0.9;
5    };
6    DecisionModel {
7      Fuzzifier { Poor; Good; Excellent; }
8      Defuzzifier { None; Limited; Full; }
9    };
10 };
```

Lines 2-5 denote the evaluation model which defines low weight on availability (0.1) and high weight on reliability (0.9). Lines 6-9 denote the decision model which is comprised of a fuzzifier, containing the membership functions: *Poor*, *Good* and *Excellent* as well as a defuzzifier, containing the membership functions: *None*, *Limited* and *Full*. It is important to emphasise that the TDS used by GREPTrust is flexible. This means that the weight ratio between the availability and reliability opinions as well the values for the membership functions within the decision model may vary in accordance to generated feedback data, job running time and threshold.

In contrast, GridPP employs an esoteric reputation model which does not consider multiple factors. Therefore, in order simulate GridPP a TDS was prepared with a single opinion (availability) equal weight rules and no decision model:

```
1  TrustDecisionStrategy {
2    EvaluationModel {
3      Availability := 1.0;
4    };
5  };
```

Recall the steps comprising the trust evaluation (feedback data generation, TDS selection and resource evaluation); the generated output is a list of resources printed

along with the recommendations made by each reputation model (GREPTrust and GridPP). In order to reveal which model tends to provide superior recommendations and the circumstances by which it does, it has been decided to execute two Grid jobs where the first job utilises the resources recommended by GridPP while the second utilises the resources recommended by GREPTrust. Each job attempts to generate a volume distribution profile based on the resources it has been recommended with.

Once both jobs complete, the two generated volume distribution profiles are compared to the original volume distribution profile (which contains 100% valid entries) in order to find out which job yields an output of higher proximity to the original volume distribution profile. If a second job's volume distribution is of higher proximity then it can be assumed that it was the result of higher quality resource selection. This serves as the benchmark criteria for the simulation process.

5.4 Simulation Design

5.4.1 Overview

The purpose of the workflow simulation is to utilise the GREPTrust testbed environment and conduct a comparative analysis based on the recommendations made by GridPP and GREPTrust, with the intention is to discover which reputation model tends to provide superior recommendations and the circumstances by which it does. The context for this comparative analysis is driven from the previously described case study where the aim was to optimise the generation of volume distribution profiles in terms of reducing the number of missing or corrupted entries in relation to the total number of entries within the file. Figure 5.6 illustrates the steps comprising the simulation process ($S_1 - S_5$) for conducting the comparative analysis.

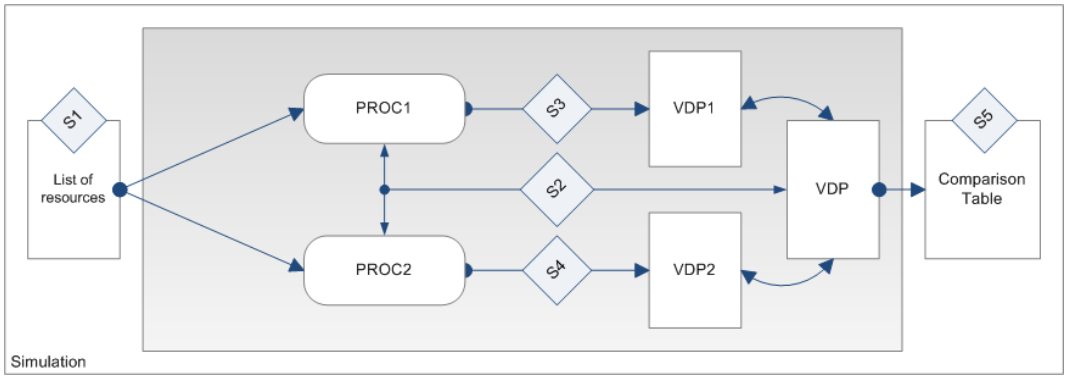


Figure 5.6: Steps comprising of the simulation process

The input for the simulation is the finite set of resources recommended by each reputation model (S_1) and a predefined threshold for controlling the final list of

selected resources (T). The simulation process constructs two correspondent Grid jobs - $PROC_1$ (GridPP) and $PROC_2$ (GREPTrust). Since this is merely a simulation task, the objective is not to generate the volume distribution profile from scratch but instead use the original volume distribution profile (VDP) as input for both $PROC_1$ and $PROC_2$. Each Grid job subdivides a copy of the original VDP between the resources it has been allocated to using round-robin scheduling algorithm and based on the probability of each resource to fail a computation cycle, it makes a decision whether to corrupt the currently held volume distribution entry or not (S_2).

Once completed, the simulation process outputs two VDP files - VDP_1 and VDP_2 generated by $PROC_1$ and $PROC_2$ respectively ($S_3 - S_4$). Both files are compared with the original VDP file in order to discover which computation yielded an output which is of higher degree of approximation (S_5). This delta comparison serves as an experiment *benchmark criteria* intended for evaluating GREPTrust and GridPP reputation models (B). The following formula provides a definition for the benchmark comparison function. It returns *true* if the percentage change between VDP_2 and VDP is smaller than the percentage change between VDP_1 and VDP, thus implicitly implying that GREPTrust has optimised resource selection.

$$f(VDP_2, VDP_1) = \frac{VDP_2 - VDP}{VDP} < \frac{VDP_1 - VDP}{VDP} \tag{5.5}$$

Table 5.5 describes the steps required for conducting a standard GREPTrust testbed experiment. Steps $(1-3)$ are part of the trust evaluation, which was already described in section 5.2.1. Steps $(4-6)$ are part of the jobs simulation. Step (4) is concerned with generating the symbol data used as input for both $PROC_1$ and $PROC_2$. It is comprised of both the symbol universe as well as the original volume distribution for each stock symbol within the universe. Step (5) is concerned with executing both volume distribution profiling jobs and storing the results to VDP_1 and VDP_2. Step (6) is concerned with comparing VDP_1 and VDP_2 with the original VDP by counting the number of missing entries, the number of corrupted entries and the number of valid entries and plotting the results into a bar chart.

GREPTrust testbed experiment		
Trust evaluation	1.	Feedback data generation
	2.	TDS selection
	3.	Resource evaluation
Jobs simulation	4.	Symbol data generation
	5.	VDP jobs execution
	6.	Comparison analysis

Table 5.5: Steps comprising of a standard GREPTrust testbed experiment

5.4.2 Jobs Simulation

Symbol data generation The symbol data generation step involves generating the symbol data used as input source for both PROC_1 and PROC_2. Figure 5.7 defines the symbol analytics data store using an entity-relationship diagram. It is comprised of five entities: *symbol, binslot, voldist, voldist_proc1* and *voldist_proc2*. The *symbol* entity contains the symbol universe - i.e. the entire list of symbols used for simulation purposes (6,466 entries). The *binslot* entity contains numbering and time slots for the 36 predefined bins (e.g. bin1 - 07:45, bin2 - 08:00 and etc). The *voldist* entity is an association of the *symbol* and *binslot* entities ($6{,}466$ symbols $\times$ 36 bins $= 232{,}776$ entries). Volume of this scale provides adequate amount of data for interpolating the number of volume distributions allocated to each Grid resource during both job executions. As a result, this greatly increases the overall probability of a calculation to fail and corrupt the bins for the currently volume held distribution.

Table 5.6 defines the mappings between the logical volume distribution profiles: VDP, VDP_1 & VDP_2 (illustrated in Figure 5.6) and their respective entity counterparts: *voldist, voldist_proc1 & voldist_proc2*. As illustrated in Figure 5.7, the *voldist_proc1* and *voldist_proc2* entities are replicas of the *voldist* entity as the *voldist* entity constitutes as the original volume distribution while *voldist_proc1* and *voldist_proc2* are intended for storing the volume distributions generated by PROC_1 and PROC_2 respectively. Therefore, each entity contains an identical set of attributes: $\{bin, SEDOL, value\}$ where *bin* is a bin identifier (range between 1-36), *SEDOL* is the symbol unique identifier and *value* is the bin value. This type of constellation serves the advantage of identically rectifying the structure and scale of the volume distribution data within the entity join: VDP $\bowtie$ VDP_1 $\bowtie$ VDP_2 thus allowing streamlined side-by-side delta comparison of the projected bin values.

Logical Entity Name	**Physical Entity Name**	**Pre Execution State**	**Post Execution State**
VDP	voldist	232,776 entries	232,776 entries
VDP_1	voldist_proc1	0 entries	232,776 entries
VDP_2	voldist_proc2	0 entries	232,776 entries

Table 5.6: Entity to table mapping including initial & post-executions states

In addition, Table 5.6 displays the pre and post execution states of the entities *voldist*, *voldist_proc1* and *voldist_proc2*. Since the *voldist* entity constitutes as an archetype, the content of it's underlying data is not influenced during the testbed life cycle (amongst job executions as well as between testbed experiments). Therefore, for efficiency purposes, the volume distribution data within *voldist* entity is merely loaded once and that is before conducting the first testbed expe-

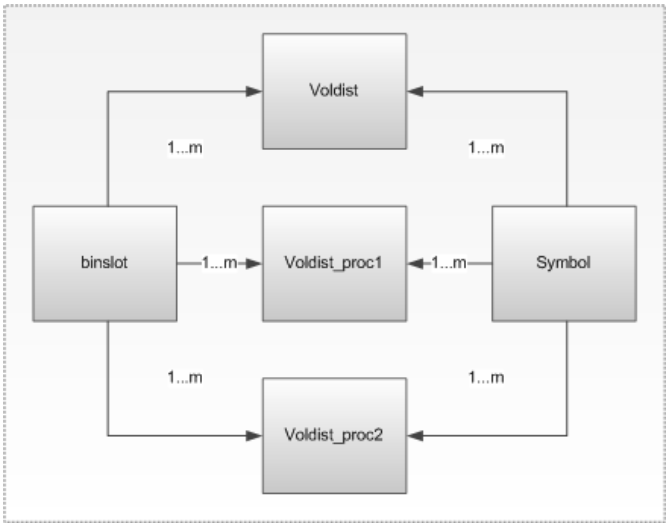


Figure 5.7: SADS - SEDOL Analytics Data Store entity-relationship diagram

riment. In contrast, the *voldist_proc1* and *voldist_proc2* entities are transient. Each job execution begins with nil entries in it's associated entity, which is gradually filled during the course of the execution. When each execution completes, the number of entries stored are equal to the number entries in the *voldist* entity ($|\ VDP_n\ | = |\ VDP\ |$; n $\in \{1,2\}$). Therefore, on simulation completion, the following cardinality rule is attained: $|VDP| = |VDP_1| = |VDP_2|$. During the tear down of a testbed experiment, the volume distribution data within these entities is truncated ($|\ VDP_n\ | = 0$; n $\in \{1,2\}$) in order to prepare for subsequent experiments.

VDP Jobs Execution The VDP jobs execution step involves submitting and executing two sequential Grid jobs - PROC_1 (GridPP) and PROC_2 (GREPTrust) with the intention to simulate the generation of two volume distribution profiles for a universe of 6,466 symbols. The resources utilised by these two jobs are based on the resource recommendations made by GridPP and GREPTrust reputation models capped by the predefined threshold function. The main principle behind the VDP job is to split symbol universe U with volume distribution scale d among n resources such that each resource r assigned to single process P operates on a dataset of size:

$$P(r_i) = \frac{Ud}{n} \text{ ,where } n \geq 0 \tag{5.6}$$

Since Ud is identical to the cardinality of VDP, the dataset allocation formula can be written also as: $P(r_i) = \frac{|VDP|}{n}$. The main purpose behind this linear interpolation is to allow equal share of data allocated between the selected resources and therefore eliminate any bias towards a particular resource. An assumption is made that all resources are idle at the initialisation of the job execution and that

processes are assigned to resources via round-robin scheduling - i.e. assign a task to the next available resource and so forth. A collection of processes $\{P(r_1),\ldots,P(r_n)\}$ constitutes as Grid job J such as $J=\{P(r_1),\ldots,P(r_n)\}$. Since this is merely a simulation activity, the objective is to concentrate of assessing the quality of the selected resources rather than concentrate on generating a new volume distribution profile with each job execution. As a result, each Process P_i iterates through it's allocated SEDOL list S_{P_i} ($\mid S \mid = \frac{U}{n}$), where for each individual SEDOL s the original volume distribution vd is loaded by resource r_i and held during time interval Δt while being computed. This process can be expressed in the following manner:

$$\begin{cases} \forall P(r_i) \in J, \text{ where } J=\{P(r_1),\ldots,P(r_n)\} \\ \quad \forall s \in S_{p_i}, \text{ where } S_{p_i}=\{S_{p_{i,1}},\ldots,S_{p_{i,j}}\} \\ \quad\quad \begin{cases} vd' := Compute(s,vd) \text{ for } \Delta t \\ Save(s,vd',table), \text{ where } table \in \{voldist\_proc1, voldist\_proc2\} \end{cases} \end{cases} \tag{5.7}$$

During the time interval Δt under the $Compute(s,vd)$ method, a *heuristic* decision is made whether to interfere with the currently held volume distribution (vd) and result in corrupting the underlying bin values:

$$vd' := Compute(s,vd) = \begin{cases} DoAvailability(vd) & \text{if } factor(m)=1; \\ DoReliability(vd) & \text{if } factor(m)=2; \\ vd & \text{otherwise;} \end{cases} \tag{5.8}$$

As illustrated, there are three possible outcomes for the $Compute(s,vd)$ method: availability interference, reliability interference or no interference (This can be enumerated as $F=\{1,2,0\}$ where F is defined as the factor identifier). Realistically, there can be additional fail factors; however - for simulation purposes, the list of fail factors is narrowed down to the ones relevant for supporting the case study (i.e. availability and reliability). In addition, in order to streamline the simulation process, an decision is made so that at most, one fail factor can result per computation at interval Δt - i.e. $[(availability \oplus reliability) \oplus none]$.

The decision on which factor to fail, is performed using a blend of stochastic and deterministic elements. The deterministic element uses the resource average feedback value for that factor (f_v) and compares it to a random generated integer $rand := random(1,100)$, such that if $f_v \leq rand$ then it is appended to the filtered factor matrix F, where the first column is the factor identifier and the second column is the factor value. The comparison is made against each defined factor (i.e. availability and reliability). As a result, the cardinality value m of matrix F is: $0 \leq m \leq 2$.

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \\ \vdots & \vdots \\ f_{m,1} & f_{m,2} \end{bmatrix} \tag{5.9}$$

This blend of stochastic and deterministic elements serves the advantage of being dependent on an historical feedback value for for determining the resiliency of a resource yet still subject to unpredicted behaviour occurrences. Consequently, the *factor*() function has three possible outcomes; no action - if the cardinality value m of the matrix F is equal to 0, the first row ($f_{[1,*]}$). Alternatively, if the $m = 1$ or select a randomised row, in case $m \geq 1$.

$$factor(m) = \begin{cases} 0 & \text{if } m = 0; \\ f_{[1,*]} & \text{if } m = 1; \\ f_{[random(1,m),*]} & \text{if } m > 1; \end{cases} \tag{5.10}$$

Figure 5.8 illustrates an archetypical (or original) volume distribution profile and compares it to scenarios where it is being corrupted due to a reliability and availability failures (utilised by the *DoReliability*() and *DoAvailability*() functions). The first chart denotes a correctly calculated volume distribution profile. As denoted by the second chart (*DoReliability()*), a simulation of reliability corruption results in the bin values being *shuffled* through a stochastic permutation of their original positions. In the third chart (*DoAvailability()*), all bin values are *scrapped* and set to the value -1, This is in order to simulate non-existing entries as a consequence of a resource not being available during the time period of the computation.

In order clarify all the previously explained steps and considerations confining the VDP jobs executions, Figure 5.9 illustrates a high level sequence diagram of a single job execution. In Step (1), the Execute(R, T) method is called, supplied with the path to the Trust Comparison Report (TCR) file and the predefined threshold. In steps (2-6), a list of resources is obtained and filtered based on the resource recommendations capped by the threshold function. In step (7), a Grid job is instantiated where in step (8) - it is invoked using the Run() method. The Grid job runs multiple processes (step (9)) where each process operates on a batch of SEDOL identifiers. The original VDP is loaded for each SEDOL in a batch (steps (10-12)) and submitted to it's target resource (step (12)), where it is altered based on the tendency of a resource to fail a computation. The GetFactor() method (step (13)) encompasses a heuristic algorithm which assigns the fail factor to the volume distribution currently held by the resource ($Failfactor \in \{Availability, Reliability, None\}$).

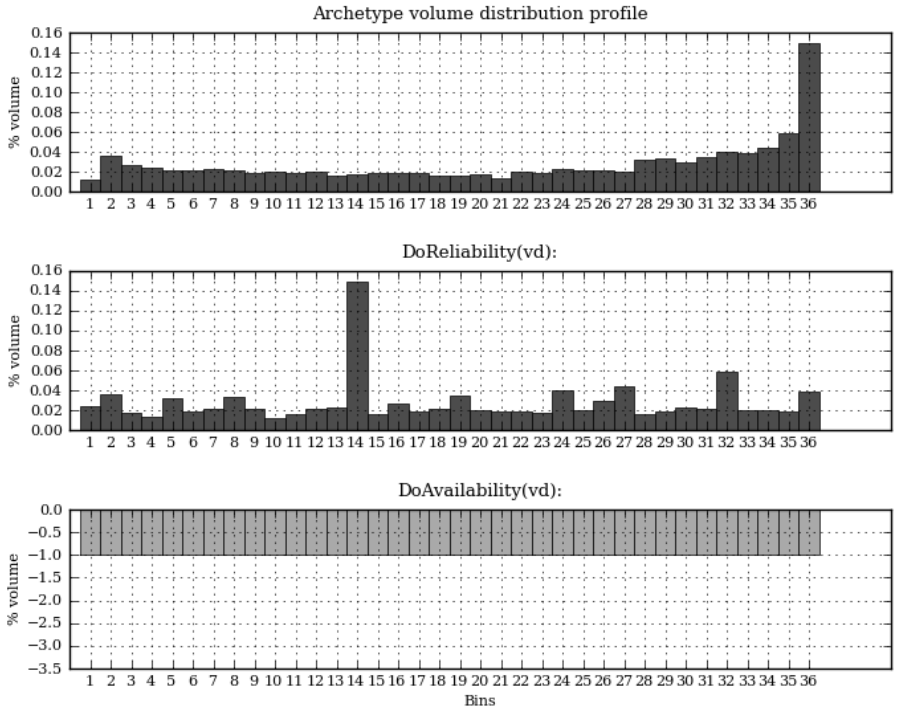


Figure 5.8: Modifications to an original volume distribution profile

The (potentially) modified volume distribution (VD') is returned by the resource to the calling process (step (14)), which saves the volume distribution entries in SADS (where $Table \in \{voldist\_proc1, voldist\_proc2\}$) (step (15)). Steps (10-15) are repeated for each SEDOL within a SEDOL array assigned to a resource by a process. Once completed, an acknowledgement is returned to the calling Grid job (step (16)). Steps (9-16) are performed in parallel for each process within the process array of a Grid job. Once completed, an acknowledgement is returned to the Execution Manager. Steps (2-17) are repeated twice (for $PROC_1$ and $PROC_2$ jobs). Once completed, an acknowledgement is returned which terminates the simulation. Appendix .3.2 contains a description of algorithms comprising of the simulation process.

As previously described, the output of the simulation yields three data sets (VDP, VDP_1 and VDP_2) containing the original volume distribution profile (VDP), the volume distribution profile generated by GridPP recommended resources (VDP_1) and the volume distribution profile generated by GREPTrust recommended resources (VDP_2). The intention of generating these data sets is allow side-by-side comparison of each of the of the volume distribution entries populated into each data set. This is in order to quantify the quality of the generated output and consecutively assess the performance of GridPP and GREPTrust reputation models.

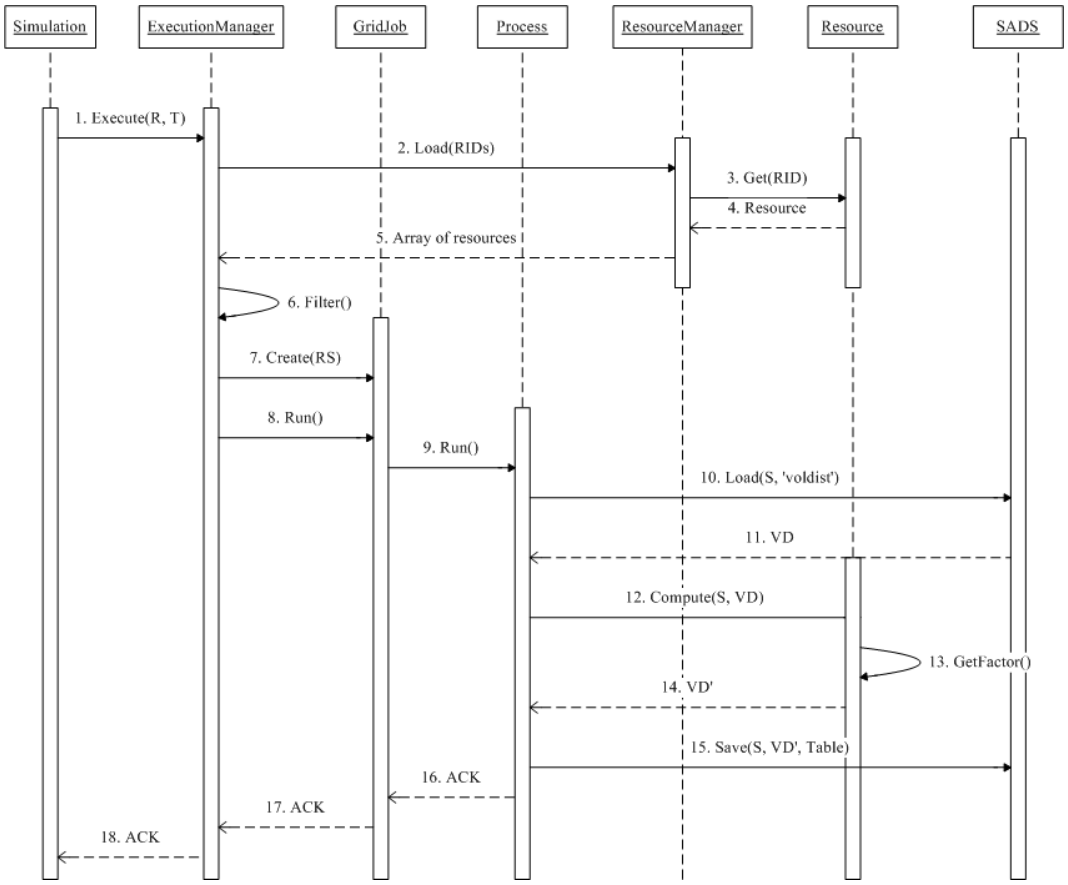


Figure 5.9: Steps comprising of the job execution process

Comparison Analysis The final step in the simulation process is the comparison analysis. This step utilises the three datasets generated as a result of the jobs execution step, by joining them together and performing a side-by-side comparison ($VDP \bowtie VDP_1 \bowtie VDP_2$), such that each entry in VDP_1 and VDP_2 is compared with the equivalent original entry in VDP. the comparison analysis algorithm maintains a collection of three arrays for both VDP_1 and VDP_2. The first array keeps a count of the number of valid entries, the second array keeps a count of the number of corrupted entries and the third array keeps a count of the missing entries.

Once the simulation completed, these two sets of arrays are compared in order to find out with execution resulted with a higher count of valid volume distribution entries. It is important to note that realistically corrupted volume distribution entries should be further analysed in order to assess the degree of corruption (i.e. the deviation of each entry from it's archetype entry). However, for the simulation purposes, all corrupted entries are assumed to be equally harmful.

5.4.3 Representation of Simulation Results

Figure 5.10 illustrates an example of VDP jobs simulation output, generated as a result of the comparison analysis algorithm. The output contains two sets of bar charts (one for each computation). The intention of this type of chart is to quantify the results of each computation by counting the number of missing entries (i.e. non-available entries), corrupted entries and valid entries. The left set represents PROC_1 results (GridPP) while the right - PROC_2 results (GREPTrust).

Each set is comprised of three bars - green, yellow and red. The green bar counts the number of valid entries whilst the yellow and red bars count the number of missing and corrupted entries respectively. The label on top of each bar indicates the number of computation cycles as well as the number volume distribution entries. A computation cycle refers to the time frame in which a particular resource holds an entire distribution for a given SEDOL S (i.e. the set of values: $S_{bin_1}(v), \ldots, S_{bin_{36}}(v)$) and a decision is being made whether to corrupt the held entries or not.

In this example, the green bar for GridPP contains the indicator: (1977|71172) meaning that 1,977 computation cycles completed successfully - that is 1,977 SEDOL entries out of 6,466 that were computed without any failures. The second indicator is the number of valid volume distribution entries (calculated as $1,977 \times 36$ bins $= 71,172$ entries) meaning that 71,172 records in dataset VDP_1 are identical to original dataset - VDP. Each computation set in the chart contains two types of scores: *standard score* and *effective score*. A standard score S_s is defined as the ratio between the number of the valid entries v and the number of total entries t:

$$S_s = \frac{v}{t} \tag{5.11}$$

As a result, GridPP generated a score of 30.58%, since $71,172/232,776 \times 100 = 30.58\%$ while GREPTrust generated a score of 71.03%. In contrast, the effective score S_e takes into account the number of non-available entries a into the calculation. This is done with consideration to the case study where non-available entries are considered benign while non-reliable (i.e. corrupt) entries are extremely harmful for the performance of the VWAP trading algorithm:

$$S_e = \frac{v+a}{t} \tag{5.12}$$

With respect to the effective score, GridPP generated the result 31.12%, since $(71,172+1,260)/232,776 \times 100 = 31.12\%$ while GREPTrust generated an effective score of 71.45%. This particular example clearly indicates a scenario where GREP-Trust *vastly outperforms* GridPP. Considering the previously described benchmark formula $f(VDP_2, VDP_1)$, the delta between the datasets VDP_2 and VDP is much smaller than the delta between VDP_1 and VDP providing higher degree of proxi-

mity to the original distribution ($\Delta VDP_2 << \Delta VDP_1$):

$$\Delta VDP_1 = \left\| \frac{71,172 - 232,776}{232,776} \right\| \times 100 = 69.425\% \tag{5.13}$$

$$\Delta VDP_2 = \left\| \frac{165,348 - 232,776}{232,776} \right\| \times 100 = 28.967\% \tag{5.14}$$

Consequently, the spreads between the scores and the effective scores are 40.46% and 40.33% respectively. These exceptional results are a combination of a constellation where availability feedbacks are ideal while reliability feedbacks are relatively low and GREPTrust considered both factors while GridPP considered merely one.

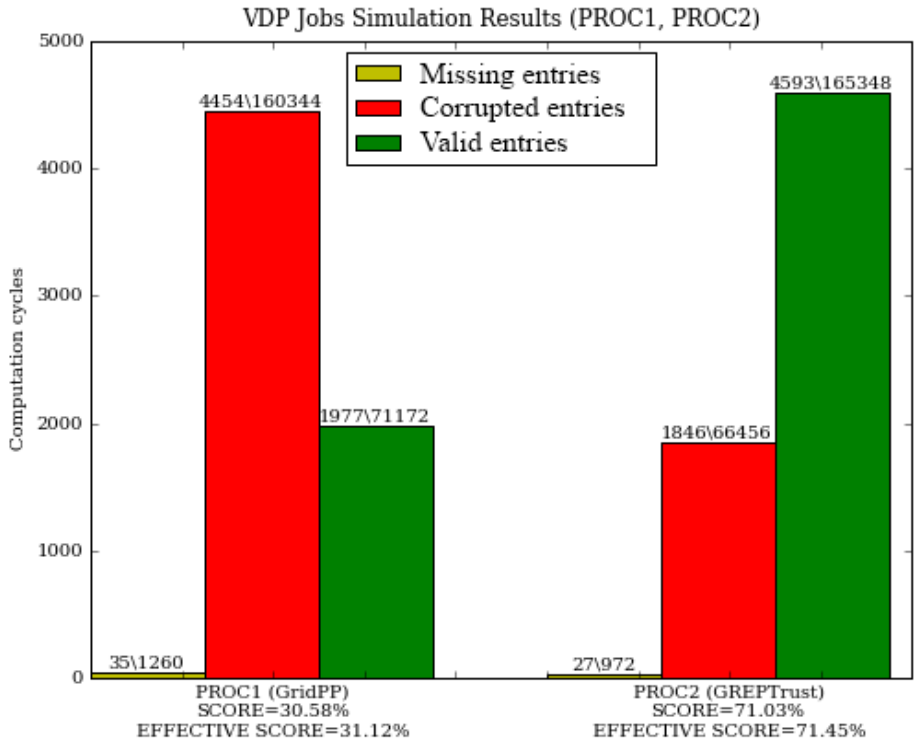


Figure 5.10: Sample VDP jobs simulation results highlighting bars in yellow - missing entries, red - corrupted entries and green - valid entries

5.5 Conclusions

This chapter encompassed three objectives. The first objective was to introduce the GREPTrust testbed architecture. It discussed the design requirements for allowing side-by-side comparison of GREPTrust exoteric reputation model with an esoteric one, such as GridPP. In addition, it observed procedures required for generating reputation model Trust Comparison Report (TCR) as well as constraints and resolutions affecting the design of the testbed.

The second objective was to present a case study, which is highly applicable to the financial industry. It focused on generating historical volume distribution profile analytics for a VWAP trading algorithm (Volume Weighted Average Price). The intention of this objective was to provide a meaningful, tangible context for evaluating the synergistic reputation-policy based trust model and consequently identify a niche area where Grid reputation-based trust management systems could be applied in order to optimise the quality of critical computations.

The third objective was to describe a workflow simulation based on the business requirements and comparison strategies identified by the second objective (case study). In particular, it identified the benchmark criteria to be applied when performing comparison analysis and described in detail each step of the simulation. The third objective finalised by presenting a sample bar chart which compares the results of the jobs based on GridPP and GREPTrust recommendations. In conclusion, the main outputs of this chapter are:

- Construction of a dedicated Grid testbed which enables the following:
    1. Deployment of GREPTrust as a Grid service and simulate interactions between Grid clients, GREPTrust and Grid resources.
    2. Infrastructure for an automated testing framework which enables comparison between exoteric and esoteric reputation models. A decision was made to compare GREPTrust reputation model with the production available GridPP reputation model.
- Identification of niche area, based on a computational finance case study where GREPTrust can potentially optimise mission critical computations by allowing adaptation for specific job requirements.
- Decision of a benchmark criteria for evaluating the performance of Grid jobs based on resources selected by GridPP and GREPTrust.

Chapter 6

Experimental Results & Post-mortem Analysis

"All life is an experiment. The more experiments you make the better."

Ralph Waldo Emerson

6.1 Introduction

The purpose of this chapter is to perform a comparative analysis of GREPTrust and GridPP reputation models based on the previously described financial case study. The general aim is to find out the conditions by which GREPTrust outperforms GridPP as well the conditions by which it does not. This is achieved by performing a series of testbed experiments followed by a post-mortem analysis.

Therefore, this chapter is divided into two sections - Experimental Results and Post-mortem Analysis. The first section describes the methodology used for conducting the testbed experiments and consecutively presents the experimental results themselves. The second section provides a discussion based on an interpretation of the experimental results and concludes over the required criteria by which GREPTrust is expected to improve resource selection in computational grids. Appendix 4 contains a dissection of the experimental results demonstrating individual test cases.

6.2 Experimental Results

6.2.1 Overview

As previously aforementioned, the GREPTrust testbed was designed with the purpose of enabling side by side comparison of esoteric and exoteric reputation

models. In the series of the presented experimental results, the GridPP esoteric reputation model is compared with GREPTrust exoteric reputation model. The evaluation of the performance of each model is derived from the results of the VDP job computation based on the set of resources recommended by each model.

The global test plan layers three batches of experiments: (i) preliminary (ii) single factor and (iii) multi factor. The preliminary experiments are essentially a set of sanity tests. They are meant to ensure that when both GREPTrust and GridPP are provided with the same TDS, the computation results are rather identical.

The single factor batch of experiments assigns both GREPTrust and GridPP reputation models to a single feedback factor (i.e. availability). This provides a fair starting point for comparison as GridPP does not consolidate multiple feedback factors when calculating reputation. The implication of this behaviour is that both models do not consider reliability feedbacks and as a result, both computations are exposed in terms of reliability failures. The TDS used by GREPTrust therefore uses a static trust evaluation model and a discretionary trust decision model.

The multi factor batch of experiments retains the static single factor TDS used by GridPP; however it expands GREPTrust to its full capability so it can be supplied with multiple factors (i.e. availability and reliability) as well as the discretionary trust decision model. This layering testing methodology allows to vigilantly observe the features of the reputation-policy based trust model by gradually increasing the set of capabilities it provides. Table 6.1 summarises the three experiment batches.

#	**Batch**	**Description**
B1	Preliminary experiments	Sanity tests. GREPTrust and GridPP are provided with an identical TDS.
B2	Single factor experiments	GREPTrust and GridPP are assigned to a single factor (availability). In addition, GREPTrust utilises a discretionary DM.
B3	Multi factor experiments	GridPP is assigned to a single factor (availability). GREPTrust utilises multiple factors (availability and reliability) as well as a discretionary DM.

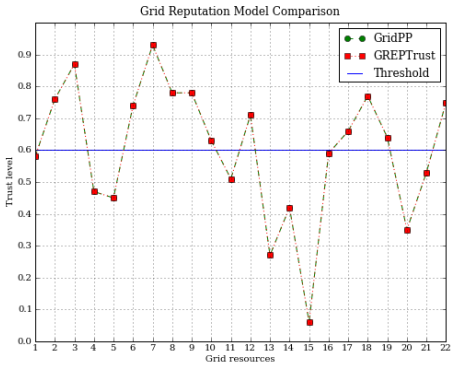
Table 6.1: Experiment batches summary

Each experiment batch is comprised of one or more scenarios. Each scenario defines different ranges for the availability and reliability feedbacks (e.g. mean value and standard deviation) and consists of one or more test cases. Each test case spreads the feedback data (within the range of the scenario) and is comprised of one

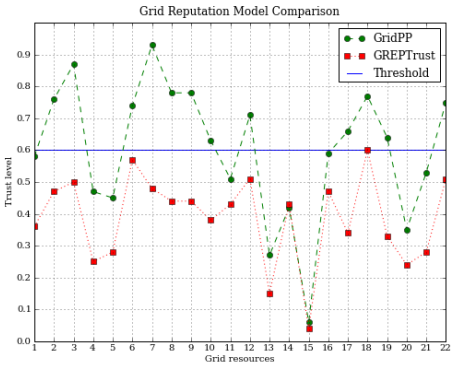
or more executions, where each execution provides a different TDS to GREPTrust. It is important to note that given the fact that each test case defines a different set of availability and feedback data, different set of resources will be selected each time. Figure 6.1 illustrates an example where two sets of resources are selected by GREPTrust in two different test cases (B3S2TC1-1A and B3S2TC4-1A). GREPTrust uses the same TDS in both test cases ($A_w = 0.5, R_w = 0.5$, no DM). However, as the feedback data differs between the two test cases, different set of resources are selected by GREPTrust each time. In this example, resources 2, 3, 6, 7, 8, 9, 10, 12, 17, 18, 19, 22 are selected in B3S2TC1-1A and resource 18 in B3S2TC4-1A. These resources were selected as they are distributed above the threshold value (0.6). Since the availability feedback data is identical in both test cases (0.6), GridPP selected the same set of resources. If the availability feedback data was different between the two test cases, GridPP would have selected a different set of resources as well.

In addition, given the VDP jobs execution, availability and feedback data also control the tendency of a resource to fail (and correspondingly corrupt the volume distribution entry as discussed in Chapter 5). The same resource can be selected in two different test cases but contribute to produce different results in case its overall accumulated availability or reliability feedback value was different between the two test cases. For example, a resource can have 0.9 for availability and 0.3 for reliability as mean feedback values for one test case and 0.9 for availability and 0.9 for reliability as mean feedback values for another test case. Suppose GREPTrust uses 0.5/0.5 availability/reliability weight ratios for its evaluation model, the same resource will be selected in both test cases (it will receive a trust value of 0.6 for the first test case and 0.9 for the second test case). The fact that the resource has a very low reliability feedback value in the first test case would increase the probability that it would corrupt a VD entry and negatively impact the produced score.

This is an example scenario where the discretionary decision model can prove useful for addressing such a limitation. The decision model can be used in a *restrictive* fashion to further narrow down the number of selected resources. This is achieved by disqualifying the selection of lower end quality resources (i.e. resources which are closer to the threshold than other resources) thus in order to find out if it would increase the results produced by GREPTrust. The surface of the restrictive decision model used throughout the experiments is illustrated in Figure 4.9 (Chapter 4). Based on the figure, a resource would need to acquire a trust value of at least 0.75 in order to be obtain the trust level of 0.6 (i.e. threshold value) and be qualified for selection. Given the previous example, a resource would need to have 0.9 for availability and at least 0.6 for reliability (instead of 0.3) in order to obtain the trust value of 0.75 and above. On the downside, this adjustment mechanism may completely block a computation as a result of disqualifying all resources in case



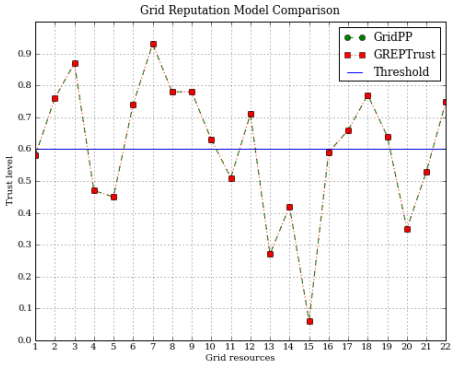
(a) Model comparison for test case execution: B3S2TC1-1A (avg. availability feedback = 0.6, avg. reliability feedback = 0.6)



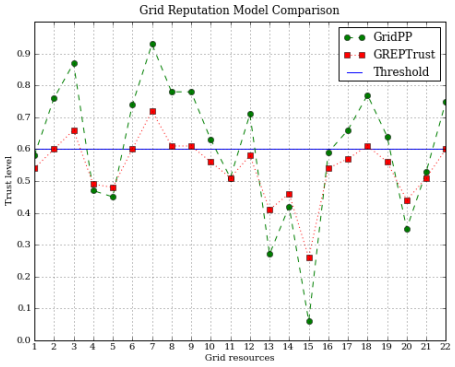
(b) Simulation results for test case execution: B3S2TC4-1A (avg. availability feedback = 0.6, avg. reliability feedback = 0.15)

Figure 6.1: Controlling resource selection using feedback data

none of the resources passed the threshold value after applying the decision model. If this is not the desirable outcome, the decision model can be made less restrictive (by changing the fuzzy membership sets comprising of the model) or completely discarding the decision model. Figure 6.2 illustrates an example where two sets of resources are selected by GREPTrust in two different executions of the same test case. The resources selected by GREPTrust in execution B2S2TC1-1A are solely driven from the evaluation model (100% weight on availability) whereas in execution B2S2TC1-1B, the restrictive decision model provides an additional layer of filtering and results in a subset of the original selection (8 resources instead of 12 resources).



(a) Model comparison for test case execution: B2S2TC1-1A (avg. availability feedback = 0.6, avg. reliability feedback = 0.6)



(b) Simulation results for test case execution: B2S2TC1-1B (avg. availability feedback = 0.6, avg. reliability feedback = 0.6)

Figure 6.2: Controlling resource selection using a restrictive Decision Model

Tables & Figures Explained

Figures 6.3 and 6.3 describe the tables and figures used throughout the different experiments.

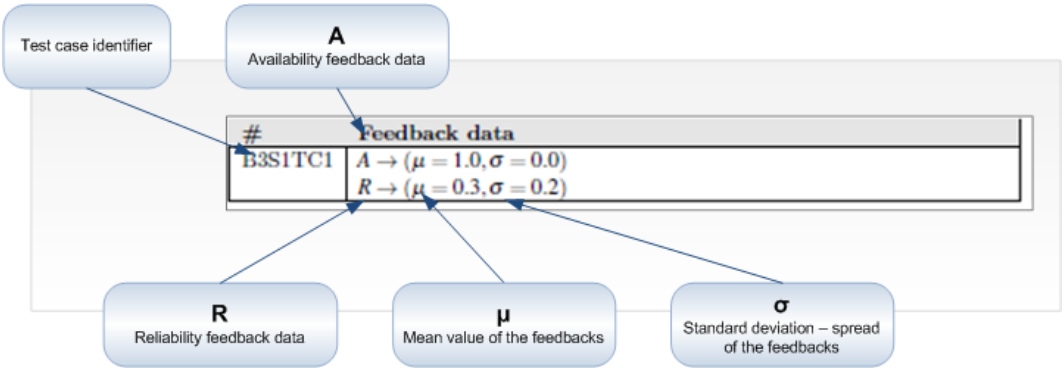


Figure 6.3: Test case table

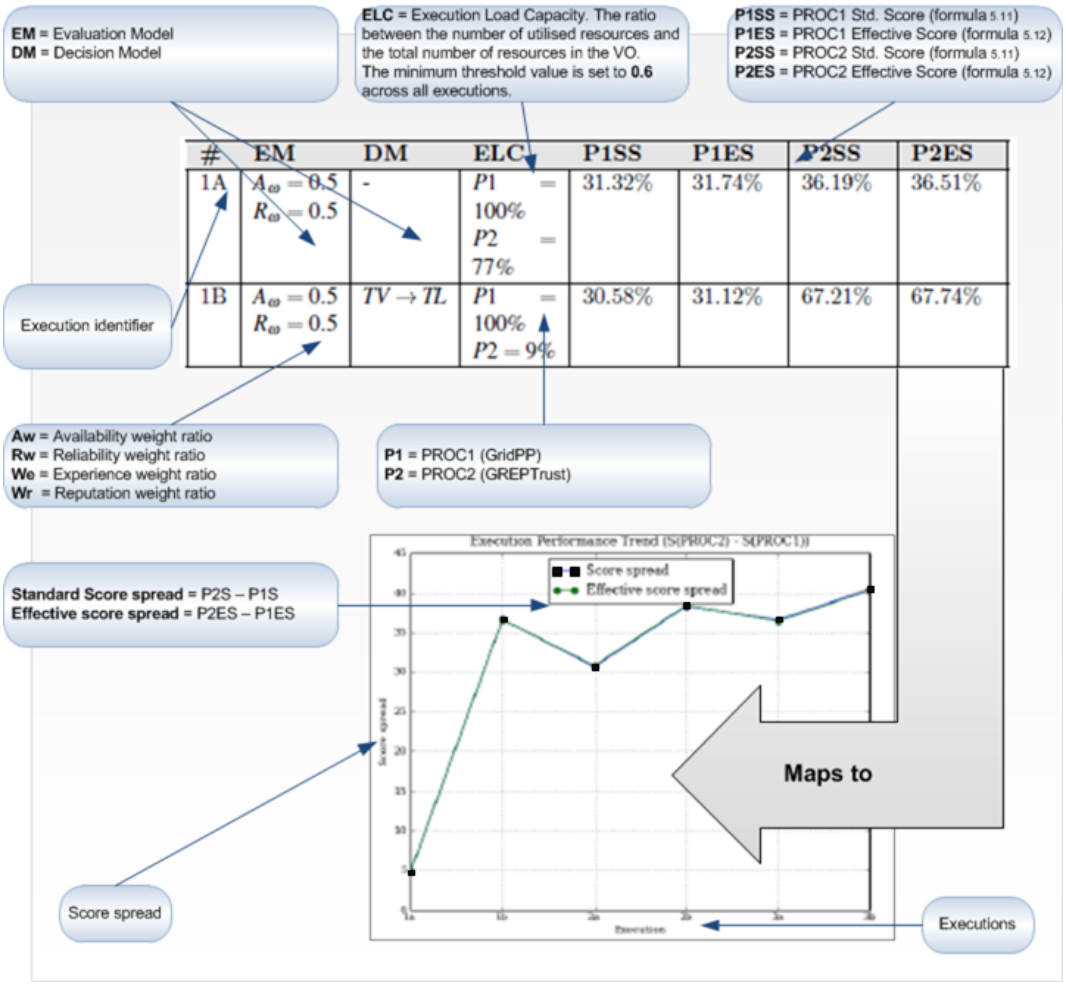


Figure 6.4: Test case executions table & trend chart

6.2.2 Batch 1: Preliminary Experiments

Overview

The purpose of this type of experiments is to ensure that both GREPTrust and GridPP behave in a similar manner when they are supplied with an identical TDS. The two models are expected to recommend an identical set of resources and consequently the two produced outputs (VDP_1, VDP_2) should receive a similar score. The TDS supplied to both models merely contains a trust evaluation model comprising of a single opinion (availability). This TDS is assigned to GridPP for the rest of the experiments. This batch includes a single scenario - B1S1 as listed in table 6.2.

#	Description
B1S1	Simultaneously increasing availability and decreasing reliability feedback data.

Table 6.2: Batch B1 scenarios

Scenario B1S1: Simultaneously increasing availability and decreasing reliability

In Scenario B1S1, the availability feedback data is gradually increased whilst simultaneously the reliability feedback data is gradually decreased. The purpose of this scenario is to demonstrate that the produced results for VDP_1 and VDP_2 should be identical regardless of the state of the feedback ratings. This scenario contains three test cases: B1S1TC1, B1S1TC2 and B1S1TC3 as listed in table 6.3.

#	Feedback data
B1S1TC1	$A \rightarrow (\mu = 0.60, \sigma = 0.1)$ $R \rightarrow (\mu = 0.60, \sigma = 0.1)$
B1S1TC2	$A \rightarrow (\mu = 0.75, \sigma = 0.1)$ $R \rightarrow (\mu = 0.45, \sigma = 0.1)$
B1S1TC3	$A \rightarrow (\mu = 0.90, \sigma = 0.1)$ $R \rightarrow (\mu = 0.30, \sigma = 0.1)$

Table 6.3: Scenario B1S1 test cases

In the current scenario, each test case is associated to a single execution (as there is no change to GREPTrusts TDS). Table 6.4 lists the executions: 1A, 1B and 1C are corresponding to test cases B1S1TC1, B1S1TC2 and B1S1TC3. The Execution Load Capacity (ELC), which informs on the percentage of utilised resources based on the recommendations of the models, is set to 59% for execution 1A and 100% for executions 1B and 1C. In execution 1A, the mean availability feedback data was

set to 0.6, which resulted in several resources falling below the threshold value (0.6) whilst in execution 1B, where the mean availability feedback data was relatively high (0.75) - all resources surpassed the threshold. This is also the case for execution 1C, where the availability feedback data was set to very high (0.9). Similarly, the standard scores for $PROC_1$ and $PROC_2$, P1SS and P2SS as well as their corresponding effective scores P1ES and P2ES are in high approximation to each other.

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 59\%$ $P2 = 59\%$	65.85%	82.97%	65.25%	82.82%
1B	$A_\omega = 1$	-	$P1 = 100\%$ $P2 = 100\%$	44.63%	57.56%	46.37%	58.52%
1C	$A_\omega = 1$	-	$P1 = 100\%$ $P2 = 100\%$	30.33%	35.25%	30.71%	35.57%

Table 6.4: Test case B1S1TC1/2/3 executions

Scenario analysis Figure 6.5 illustrates the execution trend for scenario B1S1 in which the spread (subtraction) of the standard scores (P2SS - P1SS) and the effective scores (P2ES - P21ES) for each execution are plotted as two series on a line chart. As depicted in the chart the trend for both the standard score spread and the effective score spread is approximating a plateau as the range difference in the results is around 1%. This counts as the expected behaviour for this scenario. The noticable differences between the standard and effective scores are due to the large number of missing entries for both $PROC_1$ and $PROC_2$ (appendix .4, Figure 10).

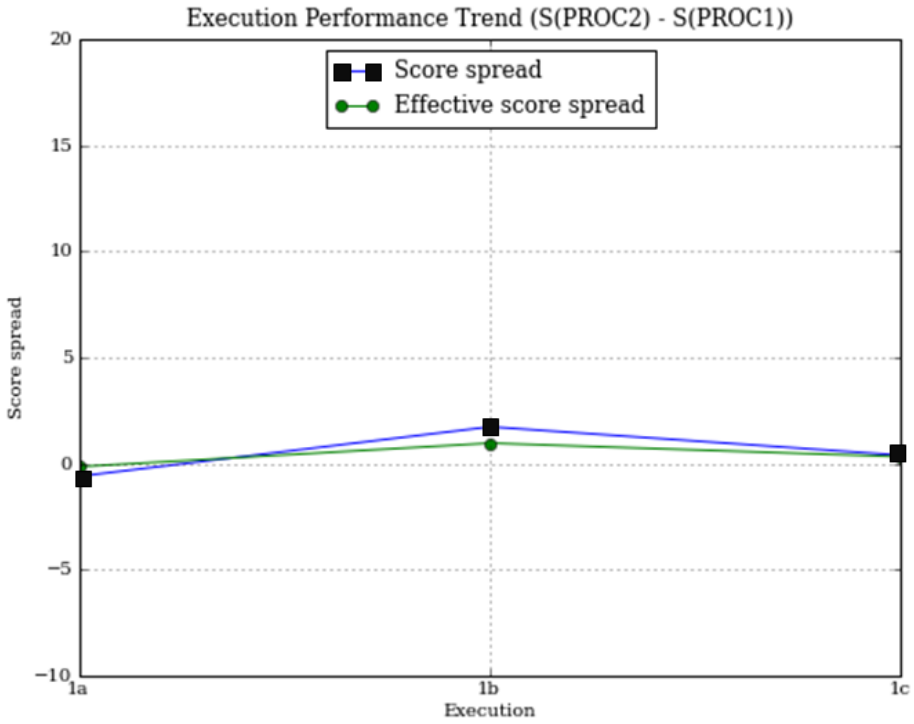


Figure 6.5: Execution trend for scenario: B1S1

6.2.3 Batch 2: Single Factor Experiments

Overview

The purpose of this type of experiments is to ensure that both GREPTrust and GridPP initiate the experiments with a fair starting point for comparison as GridPP merely relies on availability. Since GREPTrust is restrained from defining multiple opinion factors in this batch of experiments, it is left with parameters for adjusting experience and reputation ratios and a trust decision model in its reputation querying arsenal. This batch includes three scenarios as listed in Table 6.5.

#	Description
B2S1	Ideal availability & low reliability
B2S2	Consistent availability & gradually decreasing reliability
B2S3	Consistent reliability & gradually decreasing availability

Table 6.5: Batch B2 scenarios

Scenario B2S1: Ideal availability & low reliability

Scenario B2S1 deals with a situation where availability feedback data is ideal ($A_\mu = 1$) whilst reliability feedback data is low ($A_\mu = 0.3$). This scenario is selected in order to demonstrate a boundary case where GridPP and GREPTrust models are completely unaware in terms of reliability as both do not consider this factor when performing trustworthiness evaluation. This scenario contains a single test case:

#	Feedback data
B2S1TC1	$A \rightarrow (\mu = 1.0, \sigma = 0.0)$ $R \rightarrow (\mu = 0.3, \sigma = 0.2)$

Table 6.6: Scenario B2S1 test cases

Test case: B2S1TC1

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$ ($\omega_e = 0.5, \omega_r = 0.5$)	-	$P1 = 100\%$ $P2 = 100\%$	29.14%	29.14%	29.40%	29.40%
1B	$A_\omega = 1$ ($\omega_e = 0.7, \omega_r = 0.3$)	-	$P1 = 100\%$ $P2 = 100\%$	28.94%	28.94%	28.56%	28.56%
1C	$A_\omega = 1$ ($\omega_e = 0.9, \omega_r = 0.1$)	-	$P1 = 100\%$ $P2 = 100\%$	29.35%	29.35%	28.46%	28.46%

Table 6.7: Test case B2S1TC1 executions

Table 6.7 lists the executions for test case B3S1TC1. This test case is comprised of three executions, where each execution differs in terms of the experience and reputation ratio trust sources of the availability factor. A usage of a trust decision model would not be advantageous in this scenario as availability is ideal for all resources and therefore no trustworthiness scaling is needed. The ELC is identical across all test cases - 100% as all resources were qualified for selection by both GridPP and GREPTrust. Similarly, the standard scores for $PROC_1$ and $PROC_2$, P1SS and P2SS as well as their corresponding effective scores P1ES and P2ES are in high approximation to each other and therefore of no real significance.

Scenario analysis Figure 6.6 illustrates the execution trend for B2S1TC1. As depicted in the chart, there is no real trend for either the standard score spread or the effective score and therefore GREPTrust has no real advantage in this scenario. This is due to three reasons: (i) GREPTrust trust evaluation model is assigned to availability and therefore, it is exposed in terms of reliability issues, (ii) no usage of a trust decision model for the reasons previously explained and (iii) no significant advantage of adjusting experience and reputation ratio in this scenario.

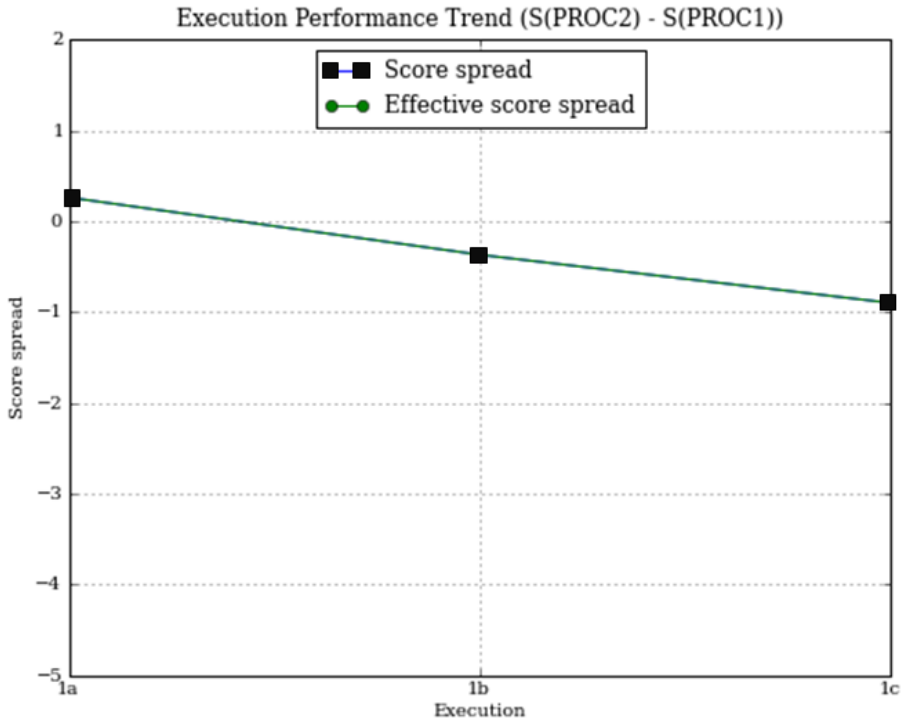


Figure 6.6: Execution trend for test case: B2S1TC1

Scenario B2S2: Consistent availability & gradually decreasing reliability

The aim of scenario B2S2 is to explore a situation in which availability feedback data remains consistent ($A_\mu = 0.6$) while reliability feedback data (R_μ) gradually decreases (0.6,...,0.15). As GREPTrust is merely assigned to availability, it can not place weight on reliability and as a result GREPTrust is not expected to have a clear advantage over GridPP. This scenario contains four test cases as listed in Table 6.8.

#	Feedback data
B2S2TC1	$A \rightarrow (\mu = 0.60, \sigma = 0.10)$ $R \rightarrow (\mu = 0.60, \sigma = 0.10)$
B2S2TC2	$A \rightarrow (\mu = 0.60, \sigma = 0.10)$ $R \rightarrow (\mu = 0.45, \sigma = 0.10)$
B2S2TC3	$A \rightarrow (\mu = 0.60, \sigma = 0.10)$ $R \rightarrow (\mu = 0.30, \sigma = 0.01)$
B2S2TC4	$A \rightarrow (\mu = 0.60, \sigma = 0.10)$ $R \rightarrow (\mu = 0.15, \sigma = 0.01)$

Table 6.8: Scenario B2S2 test cases

Test case: B2S2TC1

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 55\%$ $P2 = 55\%$	74.78%	87.81%	75.12%	87.15%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	75.32%	87.49%	79.59%	89.48%

Table 6.9: Test case B2S2TC1 executions

The produced scores by P1 and P2 are considerably different between Table 6.7 and Table 6.9. While in Table 6.7 the standard and effective scores for P1 and P2 are around 29% in Table 6.9 they sharply increase to 75%+ for the standard score (P1 and P2) and 87%+ for the effective score (P1 and P2). The sharp increase in the scores is due to a combination of two reasons: (i) ideal availability and low reliability feedback data in B2S1TC1 versus (ii) moderate availability and reliability feedback data in B2S2TC1. In a combination of ideal availability and low reliability feedback data, both GridPP and GREPTrust would qualify all resources for selection (as the TDS used by both models considers merely availability) disregarding the fact that reliability feedback data is low (0.3). As a result, during the simulations for

B2S1TC1 a large number of volume distributions were corrupted (around 165,000 entries out of 232,776 total entries) producing relatively low scores for P1 and P2. In contrast, in B2S2TC1 the availability feedback data was moderate - i.e. smaller number of resources were selected and from those resources that were selected less reliability corruptions occurred as the overall reliability data was higher (0.6).

Test case: B2S2TC2

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 55\%$ $P2 = 55\%$	50.63%	63.44%	50.40%	62.74%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	50.68%	63.16%	55.49%	65.31%

Table 6.10: Test case B2S2TC2 executions

Test case: B2S2TC3

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 55\%$ $P2 = 55\%$	29.68%	42.19%	29.21%	41.90%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	29.00%	41.79%	30.65%	40.98%

Table 6.11: Test case B2S2TC3 executions

Test case: B2S2TC4

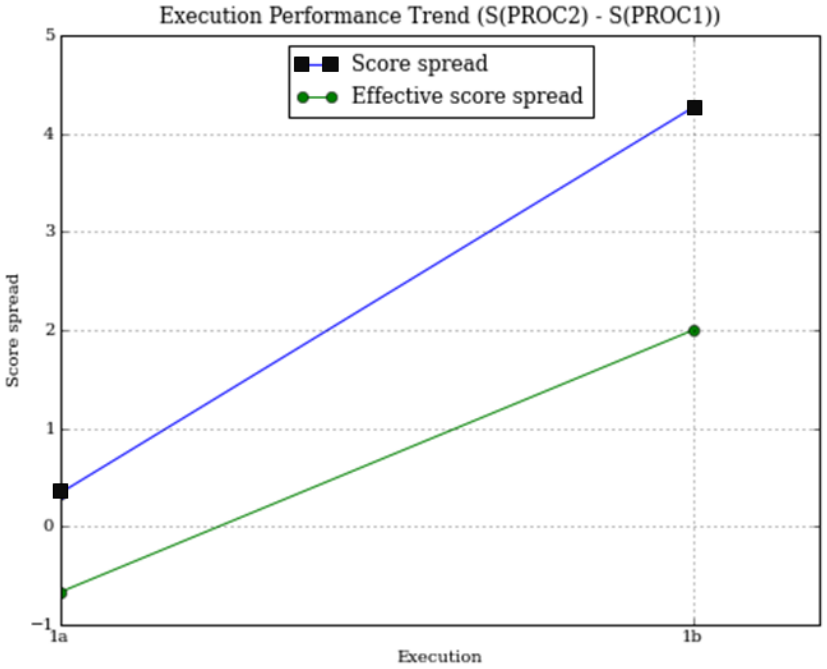
#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 55\%$ $P2 = 55\%$	16.19%	27.88%	15.90%	28.19%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	16.89%	28.95%	18.45%	28.77%

Table 6.12: Test case B2S2TC4 executions

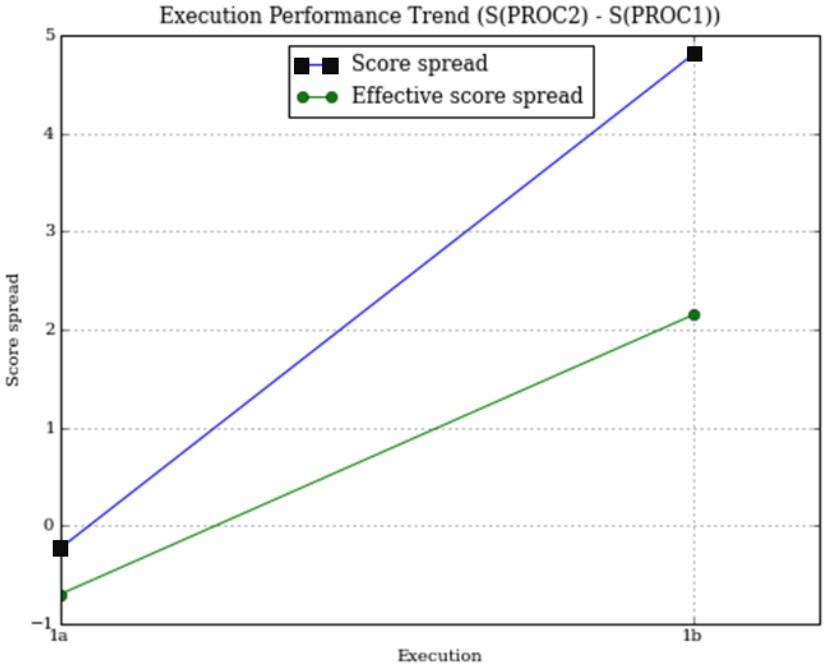
Tables 6.9, 6.10, 6.11 and 6.12 demonstrate noticable differences between the standard and effective scores. These differences are due to the large number of missing entries for both PROC_1 and PROC_2 (appendix .4, figures: 12, 13 and 14).

Scenario analysis

Figures 6.7 and 6.8 illustrate the execution trend for scenario B2S2. As depicted in both charts, GREPTrust demonstrates a slight performance advantage over GridPP. All test cases present an increase in the standard score spread (peaking at 4.81% in scenario B2S2TC2). As for the effective score, test cases B2S2TC1 and B2S2TC2 demonstrate a slight increase in the effective score while a slight decrease in test cases B2S2TC3 and B2S2TC4. The ELC has decreased to 36% for execution 1B across all test cases as a result of the restrictive trust decision model which disqualified 19% resources. To summarise, in a single factor settings a trust decision model can be used to narrow down the number of resources or alternatively expand the number of resources resulting in either a slight performance improvement (in a case of narrowing down) or a slight under performance in a case of expanding the number of resources in comparison to GridPP without any distinct advantage to any model.

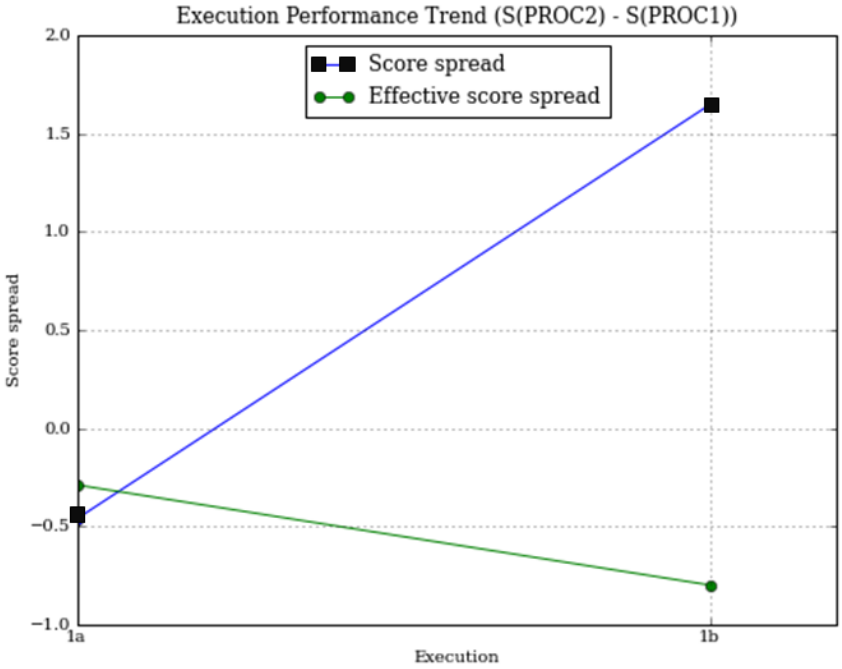


(a) Execution trend for test case: B2S2TC1

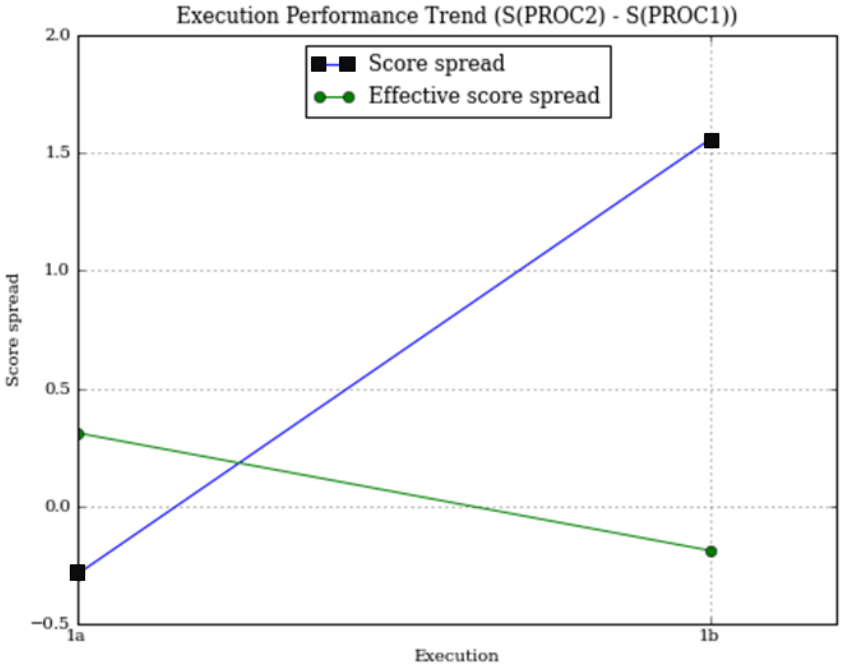


(b) Execution trend for test case: B2S2TC2

Figure 6.7: Execution trend for scenario: B2S2



(a) Execution trend for test case: B2S2TC3



(b) Execution trend for test case: B2S2TC4

Figure 6.8: Execution trend for scenario: B2S2 (continued)

Scenario B2S3: Consistent reliability & gradually decreasing availability

Scenario B2S3 attempts to explore a situation in which availability feedback data gradually decreases (0.6,...,0.15), while reliability feedback data remains consistent around the threshold value ($A_\mu = 0.6$). The purpose of this scenario is to demonstrate how GREPTrust can overcome decreasing availability feedback data using an accommodating trust decision model. In this scenario, it is expected that under certain A_μ value, GridPP would cease its recommendations completely. Similarly, due to the restrictive trust decision model used by GREPTrust, it is expected to cease recommendations as well and possibly even before GridPP does as GREPTrust is expected to filter our more resources than GridPP. This scenario contains four test cases - B2S3TC1, B2S3TC2, B2S3TC3 and B2S3TC4 as listed in Table 6.13.

#	**Feedback data**
B2S3TC1	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B2S3TC2	$A \rightarrow (\mu = 0.45, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B2S3TC3	$A \rightarrow (\mu = 0.30, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B2S3TC4	$A \rightarrow (\mu = 0.15, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$

Table 6.13: Scenario B2S3 test cases

Test case: B2S3TC1

Test case B2S3TC1 is identical to test case B2S2TC1 ($A_\mu = 0.6, R_\mu = 0.6$). Therefore, the results produced in Table 6.9 shall be reused for test case B2S3TC1.

Test case: B2S3TC2

#	**EM**	**DM**	**ELC**	**P1SS**	**P1ES**	**P2SS**	**P2ES**
1A	$A_\omega = 1$	-	$P1 = 27\%$ $P2 = 27\%$	60.73%	77.53%	61.85%	78.13%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 27\%$ $P2 = 9\%$	61.49%	76.97%	60.98%	72.92%

Table 6.14: Test case B2S3TC2 executions

Test case: B2S3TC3

In Table 6.15, the ELC remained at 5% for P1 and P2 across executions 1A and 1B. In execution 1A, a single resource (ID=14) was qualified for the computation receiving a mutual trust level value of 0.74 by GridPP and GREPTrust. In execution 1B, the same resource received a trust level value of 0.74 by GridPP and 0.6 by GREPTrust due to the restrictive trust decision model. As both values still meet the threshold requirements, the same resource was qualified by both GridPP and GREPTrust in scenario 1B and as a result the ELC factor remained identical.

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1 = 5\%$ $P2 = 5\%$	42.45%	55.35%	41.77%	55.43%
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1 = 5\%$ $P2 = 5\%$	41.66%	54.55%	42.05%	55.18%

Table 6.15: Test case B2S3TC3 executions

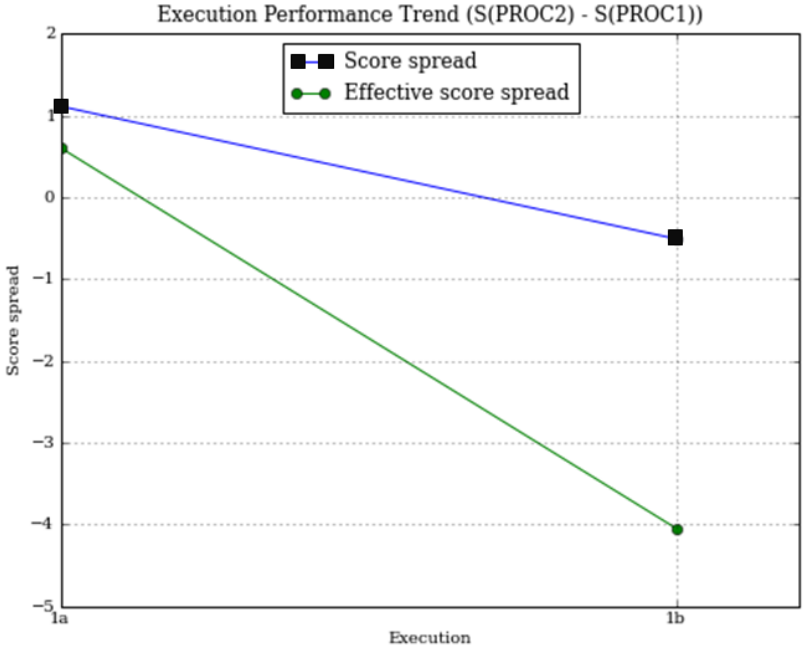
Tables 6.14 and 6.15 demonstrate substantial differences between the standard and effective scores. These differences are due to the large number of missing entries for both PROC_1 and PROC_2 (appendix .4, figures: 15 and 16).

Test case: B2S3TC4

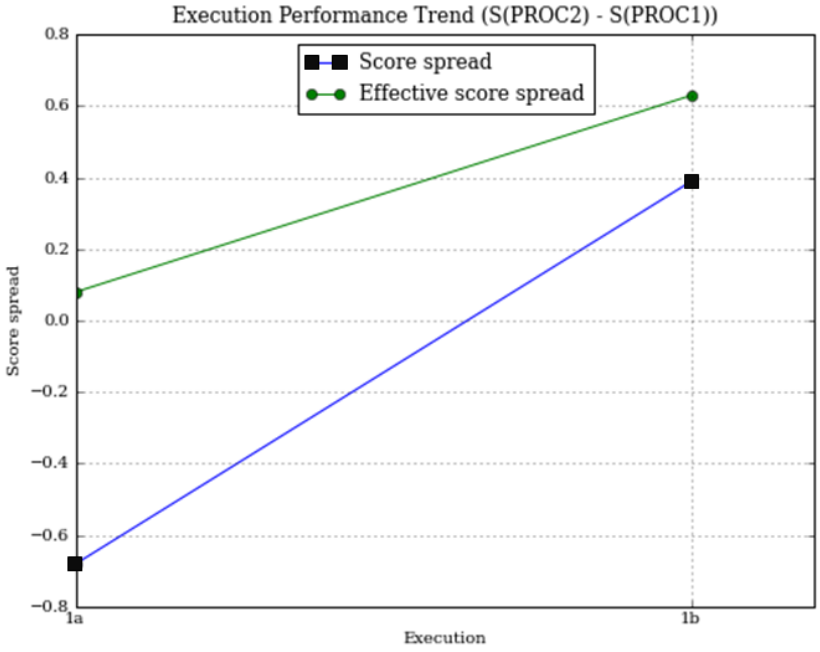
In Table 6.16 neither GridPP nor GREPTrust could qualify any resource for selection. This is due to an extremely low availability feedback data ($A_\mu = 0.15$) GridPP which resulted in all resources trust levels to set below the threshold value (0.6).

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 1$	-	$P1, P2 =$ 0%	-	-	-	-
1B	$A_\omega = 1$	$TV \rightarrow TL$	$P1, P2 =$ 0%	-	-	-	-

Table 6.16: Test case B2S3TC4 executions



(a) Execution trend for test case: B2S3TC2



(b) Execution trend for test case: B2S3TC3

Figure 6.9: Execution trend for scenario: B2S3

Scenario analysis

Figure 6.9 illustrates the execution trend for scenario B2S3. Similar to scenario B2S2, GREPTrust does not demonstrate a real performance advantage over GridPP This results in an approximately 4.27% spread in test case B2S3TC1, execution 1B to approximately -0.51% spread in test case B2S3TC2, execution 1B despite the usage of a restrictive trust decision model which narrowed down resources.

Througout scenario B2S3, the standard and effective scores produced in Tables 6.9 and 6.14- 6.16 were continuously decreased until discontinued in Table 6.16. The decrease in the results is not related to the decrease of the ELC but rather to the deterioration of the availability feedback value in each test case. This scenario concludes with slight performance advantage to GREPTrust in scenarios B2S3TC1 and B2S3TC3 as well as slight under performance in scenario B2S3TC2.

6.2.4 Batch 3: Multi Factor Experiments

Overview

The purpose of this type of experiments is to evaluate GREPTrust performance when its trust evaluation model is eligible to support multiple opinion factors, such as availability and reliability whilst GridPP remains assigned to availability. This constellation has great importance with respect to the requirements of the case study as opinion weights can be adjusted to reflect a trade off between a desirable computation (i.e. minimum number of corrupt entries) and execution load capacity. This batch includes three complementing scenarios to 6.5, as listed in Table 6.17.

#	Description
B3S1	Ideal availability & low reliability
B3S2	Consistent availability & gradually decreasing reliability
B3S3	Consistent reliability & gradually decreasing availability

Table 6.17: Batch B3 scenarios

Scenario B3S1: Ideal availability & low reliability

Scenario B3S1 is an expansion of scenario B2S1. Given an ideal availability feedback data ($A_\mu = 1$) and low reliability feedback data, GREPTrust has a clear advantage of placing weight on reliability such that ($A_\omega \leq R_\omega$) and consequently optimise the computation. In contrast, GridPP is expected to assume that all resources are valid for selection as it would completely ignore the reliability factor. The reason for including this boundary case scenario is to reveal the maximum spread GREPTrust can achieve over GridPP. This scenario contains a single test case - B3S1TC1:

#	Feedback data
B3S1TC1	$A \rightarrow (\mu = 1.0, \sigma = 0.0)$ $R \rightarrow (\mu = 0.3, \sigma = 0.2)$

Table 6.18: Scenario B3S1 test cases

Table 6.19 lists the executions for test case B3S1TC1. This test case is comprised of six executions, where in each execution the pool of resources for $PROC_2$ is throttled by either increasing the reliability weight (in the expense of the availability weight) or by placing a trust decision model with restrictive trustworthiness scaling. The purpose of choosing this set of executions is to demonstrate the process of resource selection optimisation. This is achieved by initially placing an equal ratio of availability and reliability opinions (1A) and then gradually increasing the reliability weight until reaching 0.9 (3B). Any attempt to additionally increase the reliability weight ratio would result in GREPTrust disqualifying all VO resources as none of the selected resources would have passed the threshold value for the execution.

Test case: B3S1TC1

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 100\%$ $P2 = 64\%$	29.60%	29.60%	40.63%	40.63%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 100\%$ $P2 = 18\%$	29.01%	29.01%	56.91%	56.91%
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 100\%$ $P2 = 23\%$	28.55%	28.55%	54.08%	54.08%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 100\%$ $P2 = 5\%$	28.67%	28.67%	72.73%	72.73%
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 100\%$ $P2 = 5\%$	29.06%	29.06%	74.11%	74.11%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 100\%$ $P2 = 5\%$	29.88%	29.88%	74.05%	74.05%

Table 6.19: Test case B3S1TC1 executions

Scenario analysis Figure 6.10 illustrates the execution trend for test case B3S1TC1. As depicted in the chart, the trend for both the standard score spread and the effective score spread is continually increasing which implies that GREPTrust is vastly outperforming GridPP. The standard and the effective score spread series: 1A - 11.09%, 1B - 27.90%, 2A - 25.53%, 2B - 44.06%, 3A - 45.05%, 3B - 44.17% demonstrate clear advantage to GREPTrust. This behaviour is the result of two predominant factors: (i) continuous weight balancing between availability and reliability (gradually increasing reliability weight over availability weight) and (ii) usage of a restrictive trust decision model which provides an additional layer of filtering (in 1B, 2B and 3B). At peak settings (execution 3A) GREPTrust manages to outperform GridPP by approximately 45% for the spread of both the standard score and the effective score. This counts as paramount improvement over scenario B2S1.

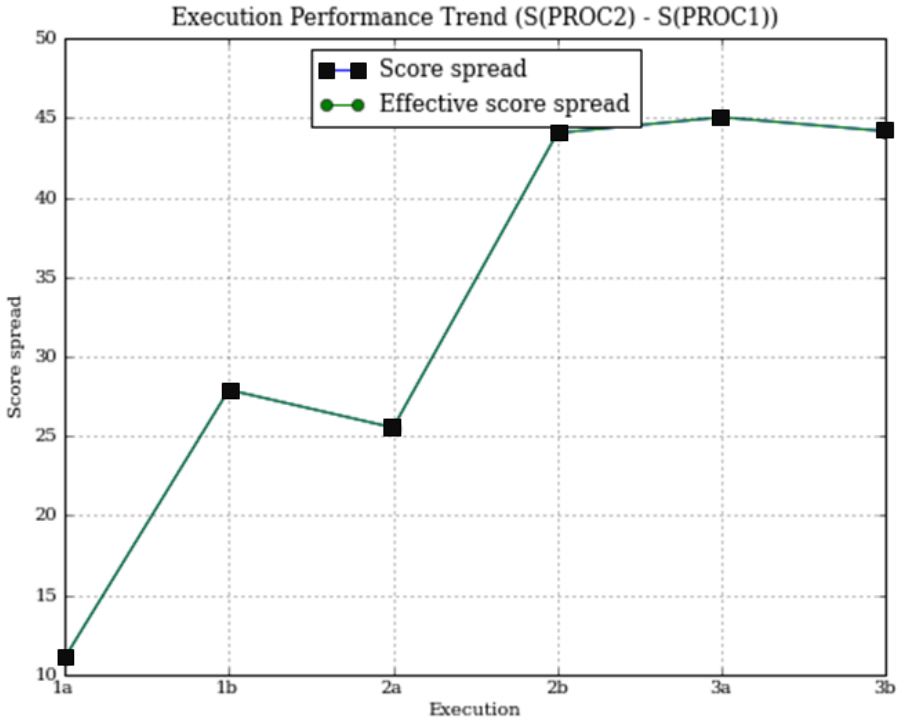


Figure 6.10: Execution trend for test case: B3S1TC1

Scenario B3S2: Consistent availability & gradually decreasing reliability

Scenario B3S2 is a multi factor expansion of scenario B2S2. The aim of this scenario is to explore a situation in which availability feedback data remains consistent ($A_\mu = 0.6$) while reliability feedback data (R_μ) gradually decreases ($0.6, \ldots, 0.15$). It is expected that the greater the feedback delta between availability and reliability is, the more likely for GREPTrust to outperform GridPP as it can place heavier weight on reliability (in the expense of availability weight) and as a result filter out as much as possible non-reliable resources. The performance of GridPP is expected to decrease as reliability feedback data decreases. This scenario contains four test case - B3S2TC1, B3S2TC2, B3S2TC3 and B3S2TC4 as listed in Table 6.20.

#	**Feedback data**
B3S2TC1	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B3S2TC2	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.45, \sigma = 0.20)$
B3S2TC3	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.30, \sigma = 0.20)$
B3S2TC4	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.15, \sigma = 0.20)$

Table 6.20: Scenario B3S2 test cases

Test case: B3S2TC1

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 55\%$ $P2 = 55\%$	76.06%	87.95%	76.18%	87.81%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	74.79%	87.29%	80.56%	90.40%
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 55\%$ $P2 = 55\%$	74.62%	87.07%	74.65%	87.12%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	75.09%	87.70%	80.33%	89.79%
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 55\%$ $P2 = 55\%$	75.58%	87.61%-	76.00%	88.18%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 36\%$	74.93%	87.21%	79.45%	90.01%

Table 6.21: Test case B3S2TC1 executions

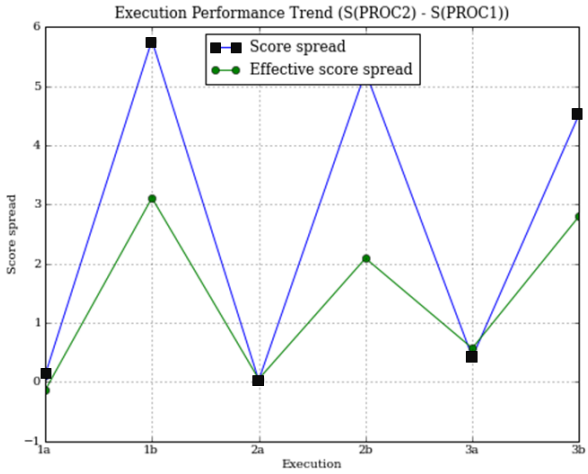


Figure 6.11: Execution trend for test case: B3S2TC1

Test case: B3S2TC2

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 55\%$ $P2 = 41\%$	50.74%	63.30%	58.23%	72.02%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 9\%$	49.04%	61.97%	70.82%	82.77%
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 55\%$ $P2 = 27\%$	50.71%	63.49%	62.50%	79.43%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 5\%$	50.67%	62.48%	73.41%	85.06%
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 55\%$ $P2 = 27\%$	49.74%	62.37%-	60.42%	84.49%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 9\%$	49.57%	62.68%	60.86%	89.07%

Table 6.22: Test case B3S2TC2 executions

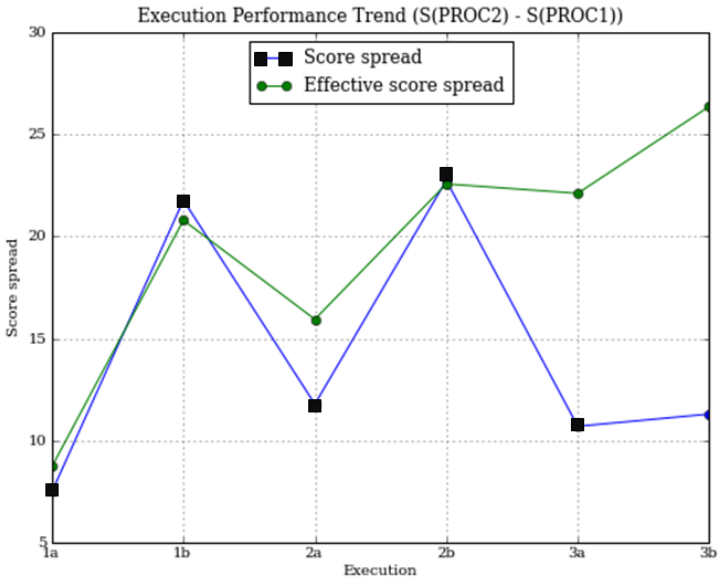


Figure 6.12: Execution trend for test case: B3S2TC2

Test case: B3S2TC3

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 55\%$ $P2 = 14\%$	29.54%	42.19%	50.71%	59.46%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 55\%$ $P2 = 18\%$	29.57%	42.07%	48.55%	66.07%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 55\%$ $P2 = 5\%$	29.82%	42.34%	41.25%	87.01%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-

Table 6.23: Test case B3S2TC3 executions

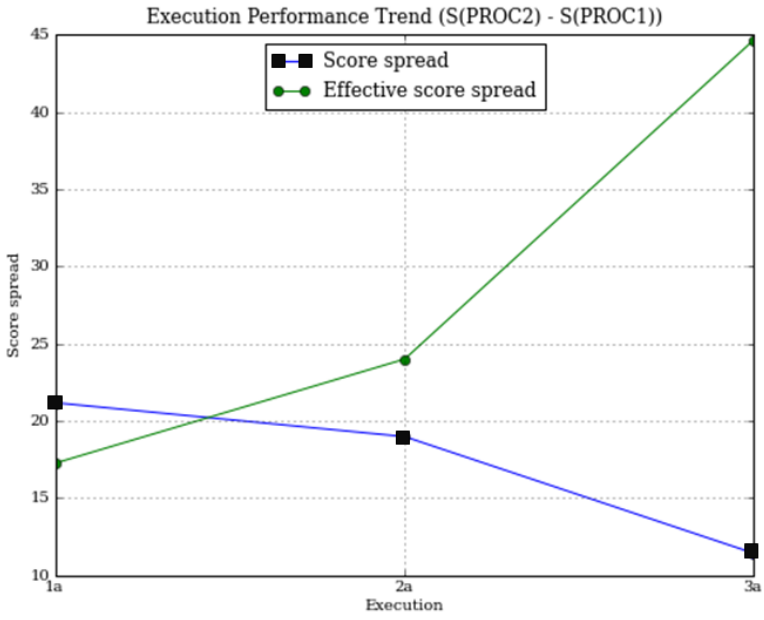


Figure 6.13: Execution trend for test case: B3S2TC3

Test case: B3S2TC4

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 55\%$ $P2 = 5\%$	15.91%	28.30%	41.51%	53.03%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 55\%$ $P2 = 0\%$	-	-	-	-

Table 6.24: Test case B3S2TC4 executions

In Table 6.21, despite decreasing the availability weight whilst increasing the reliability weight the P2 ELC factor remained at 55% for executions *A and 36% for exections *B. Moreover, the scores for P1 and P2 remained relatively similar across all executions. This is due to having identical reliability and availability feedback data. In such a circumstance, weight adjustments do not make any difference.

In Table 6.22, the scores for P2 spike throughout executions *B. This is due to a restrictive trust decision model which was added on to the evaluation model set throughout executions *A. The restrictive trust decision model filtered out lower ranking resources in each *B execution and resulted in an improved overall score. In addition, the standard score for P2 is considerably smaller in 3B comparing to its produced standard scores in 1B and 2B. In execution 3B, two resources were selected by GREPTrust - 11 ($A_\mu = 0.51, R_\mu = 0.82$) and 12 ($A_\mu = 0.71, R_\mu = 0.74$) whereas in execution 2B for example, a single resource was selected - 2 ($A_\mu = 0.76, R_\mu = 0.73$). As previously described, the mean feedback value controls the tendency of a resource to corrupt a volume distribution entry during a computation. Since resource 11 has considerably lower availability feedback ratings, it contributed to the lower score produced by P2 in execution 3B. This also explains the relatively high effective score in 3B as the effective score counts also the number of missing entries.

Scenario analysis

Figures 6.11 - 6.13 illustrate the execution trend for scenario B3S2. As anticipated, the bigger the delta between reliability and availability feedback data, the greater the score spread between GREPTrust and GridPP. In test cases B3S2TC1 and B3S2TC2 the standard score spread peaked between approximately 5.77% (B3S2TC1, 1B) and 22.74% (B3S2TC2, 2B). In contrast, in test cases B3S2TC3 and B3S2TC4 the score spread climbed to a peak range of approximately 21.17% (B3S2TC4, 3B) and 25.6% (B3S2TC3, 1A). The effective score spread peaked at a range of 26.39% (B3S2TC2, 3B) and 44.67% (B3S2TC4, 3A) counting as paramount improvement over scenario B2S2. In addition, the utilisation of a restricting trust decision model ensured that only resources with higher reliability rankings would be selected. As a result, several executions were completely disqualified by GREPTrust. This is particularly notable in test case B2S2TC4 where only execution 1A was allowed by GREPTrust.

Scenario B3S3: Consistent reliability & gradually decreasing availability

Scenario B3S3 is a multi factor expansion of scenario B2S3. This scenario attempts to explore a situation in which availability feedback data gradually decreases while reliability feedback data remains consistent around the threshold value ($A_\mu = 0.6$). The purpose of this scenario is to conclude over a conflict of interest condition - GREPTrust aims to place as much weight as possible on reliability yet it has to consider the decreasing availability. It is expected that the more weight GREPTrust places on reliability (in the expense of availability) the smaller the performance would be. In addition, the smaller availability value the lesser the chance that GridPP would qualify any resource for a computation. It is expected that under a certain A_μ value, GridPP would cease its recommendations completely. GREPTrust, on the other hand could refer to either placing more weight on reliability or using an accommodating trust decision model as a work around. This scenario contains four test cases - B3S3TC1, B3S3TC2, B3S3TC3 and B3S3TC4, as listed in Table 6.25.

#	**Feedback data**
B3S3TC1	$A \rightarrow (\mu = 0.60, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B3S3TC2	$A \rightarrow (\mu = 0.45, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B3S3TC3	$A \rightarrow (\mu = 0.30, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$
B3S3TC4	$A \rightarrow (\mu = 0.15, \sigma = 0.20)$ $R \rightarrow (\mu = 0.60, \sigma = 0.20)$

Table 6.25: Scenario B3S3 test cases

Test case: B3S3TC1

Test case B3S3TC1 is identical to test case B3S2TC1 ($A\mu = 0.6, R\mu = 0.6$). Therefore, the results produced in Table 6.21 shall be reused for test case B3S3TC1.

Test case: B3S3TC2

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 27\%$ $P2 = 41\%$	60.25%	76.40%	58.34%	85.97%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 27\%$ $P2 = 9\%$	61.06%	76.38%	72.02%	88.86%
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 27\%$ $P2 = 45\%$	60.38%	76.34%	55.44%	85.91%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 27\%$ $P2 = 23\%$	60.52%	76.59%	59.68%	91.22%
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 27\%$ $P2 = 50\%$	61.09%	76.21%	54.19%	88.71%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 27\%$ $P2 = 27\%$	60.42%	77.23%	57.79%	90.91%

Table 6.26: Test case B3S3TC2 executions

Test case: B3S3TC3

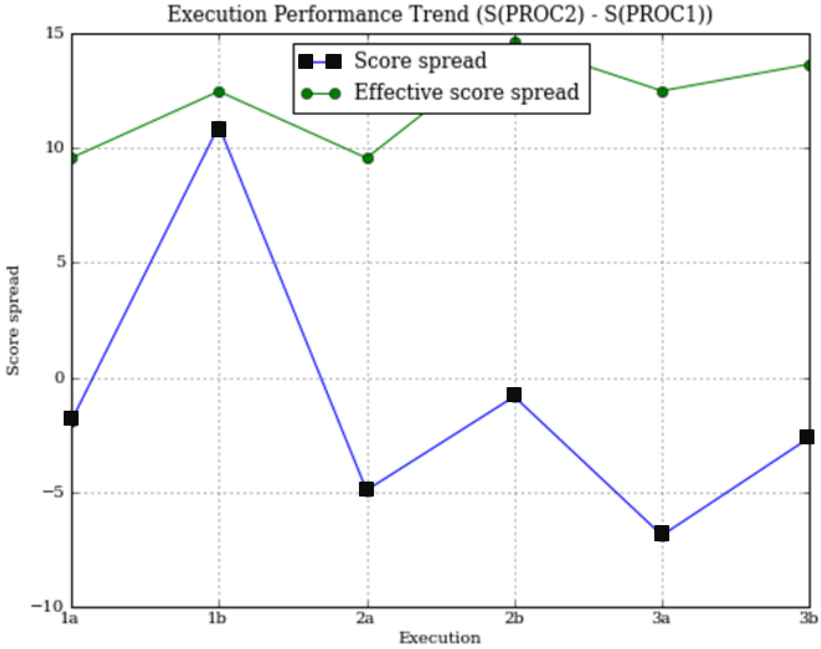
#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 5\%$ $P2 = 14\%$	40.84%	54.21%	50.87%	90.64%
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 5\%$ $P2 = 0\%$	-	-	-	-
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 5\%$ $P2 = 27\%$	42.02%	55.24%	41.40%	90.30%
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 5\%$ $P2 = 5\%$	41.88%	54.53%	46.07%	96.8%
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 5\%$ $P2 = 50\%$	41.63%	54.64%	31.18%	88.35%
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 5\%$ $P2 = 18\%$	42.17%	54.92%	43.61%	91.63%

Table 6.27: Test case B3S3TC3 executions

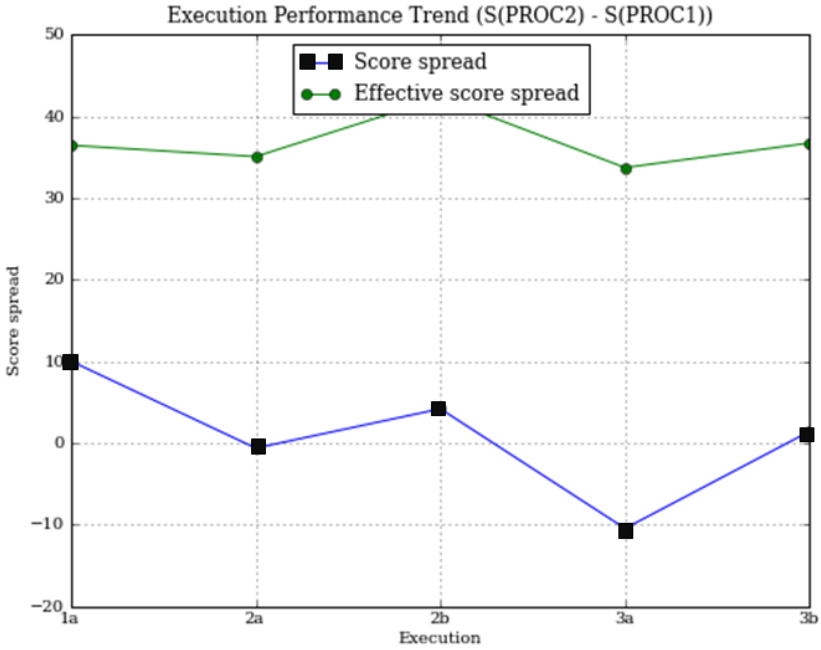
Test case: B3S3TC4

#	EM	DM	ELC	P1SS	P1ES	P2SS	P2ES
1A	$A_\omega = 0.5$ $R_\omega = 0.5$	-	$P1 = 0\%$ $P2 = 5\%$	-	-	-	-
1B	$A_\omega = 0.5$ $R_\omega = 0.5$	$TV \rightarrow TL$	$P1 = 0\%$ $P2 = 0\%$	-	-	-	-
2A	$A_\omega = 0.3$ $R_\omega = 0.7$	-	$P1 = 0\%$ $P2 = 23\%$	-	-	-	-
2B	$A_\omega = 0.3$ $R_\omega = 0.7$	$TV \rightarrow TL$	$P1 = 0\%$ $P2 = 0\%$	-	-	-	-
3A	$A_\omega = 0.1$ $R_\omega = 0.9$	-	$P1 = 0\%$ $P2 = 41\%$	-	-	-	-
3B	$A_\omega = 0.1$ $R_\omega = 0.9$	$TV \rightarrow TL$	$P1 = 0\%$ $P2 = 9\%$	-	-	-	-

Table 6.28: Test case B3S3TC4 executions



(a) Execution trend for test case: B3S3TC2



(b) Execution trend for test case: B3S3TC3

Figure 6.14: Execution trend for scenario: B3S3

Scenario analysis

Figure 6.14 illustrate the execution trend for scenario B3S3. In this scenario, GREPTrust still mostly outperforms GridPP; however its performance is moderate in comparison to the previous two scenarios resulting in a peak of 10.96% for the standard score spread in B3S3TC2, execution 1B. In B3S3TC3 execution 3A, GREPTrust considerably underperforms against GridPP where the standard score spread is -10.45%. This affirms a situation of conflict of interest where GREPTrust places extensive weight on reliability (0.9) in order to address the VDP generation job requirements yet it disregards the low availability feedback data (0.3) which is considered by GridPP. In contrast to the dwindling standard score spread, the effective score spread was completely in the advantage of GREPTrust. The spread climbed until it peaked at 42.27% in B3S3TC3 execution 2B. These results are a consequence of a combination of a continuously decreasing availability feedback data while GREPTrust consistently increasing the weight of reliability.

An interesting observation is made in test case B3S3TC4. As mean availability feedback data was set to extremely extremely low ($A_\mu = 0.15$) GridPP could not qualify any resources for selection as there was no resource of which its availability value was greater than the threshold value (0.6). GREPTrust, on the other hand, qualified compute resources in 4 out of the 6 executions (1A, 2A, 3A and 3B). This is a consequence of continuously increasing the reliability weight which is fairly high (0.6) in the expense of the availability weight. Potentially, GREPTrust could have qualified resources throughout all executions by using a less restrictive trust decision model. As a result, this scenario demonstrates that GREPTrust trust model is flexible and allows adjustments to made dynamically while GridPP remains rigid. This scenario concludes with a *qualitative* performance advantage to GREPTrust.

6.3 Post-mortem Analysis

6.3.1 Discussion

In order to evaluate the overall performance of GREPTrust and GridPP reputation models, Tables 6.29 and 6.30 were created with the intention of summarising the ranges of standard score and effective score spreads. These results are plotted in Figure 6.15. For each scenario (SS1-SS3, ES1-ES3), the range between the minimum spread and the maximum spread is displayed for both the single factor settings and the multi factor settings (i.e. B2 and B3 respectively). As can be concluded from both tables, substantial improvements are demonstrated in the multi factor batch of experiments (scenarios SS1-SS2 and ES1-ES2) resulting in a *quantitative* advantage for GREPTrust peaking at additional 45.05% valid entries (scenarios SS1, ES1), 25.6% and 44.67% valid entries (scenarios SS2 and ES2 respectively) in the generated volume distribution profile. This is a consequence of gradually increasing the reliability weight factor inside GREPTrust trust evaluation model as well as a usage of a restrictive trust decision model. As GridPP was consistently assigned to merely availability, it had been unaware of low reliability resources when making resource recommendations and therefore resulted in lower quality computations.

Reasonable performance improvements are also noticeable in the multi batch scenarios SS3, ES3 (10.96% and 42.27% respectively); however the main advantage for GREPTrust in these scenarios is *qualitative* - as availability feedback data decreased, GridPP recommended fewer resources until ceased recommendations completely. GREPTrust on the other hand, utilised its trust decision model to dynamically expand or narrow down the number of qualified resources and blocked fewer computations. In contrast, in the single batch experiments (SS1 - SS3 and ES1 - ES3), GREPTrust did not demonstrate substantial advantage over GridPP. This is despite the usage of a restrictive trust decision model which narrowed down the number of selected resources (in comparison to the number of selected resources by GridPP).

With consideration to the VDP case study, GREPTrust has demonstrated the capability of substantially optimising the quality of the daily generated volume distribution profile. However, it is important to stress that optimal results are depended on intrinsic knowledge of the reputation data which is required to generate an optimised TDS. As a repercussion of the analysis of the experiment results, GREPTrust support of characterisation and modulation of trust are particularly noticeable in the multi-factor experiments. In conclusion, GREPTrust has proven useful for situations where granular (i.e. detailed) selection is needed. For example in grids where there is a large delta between quality factors. On the other hand, in grids where quality factors are unified, GridPP would offer a more simplified reputation service eliminating the burden of multiple QoS factors and complex decision making.

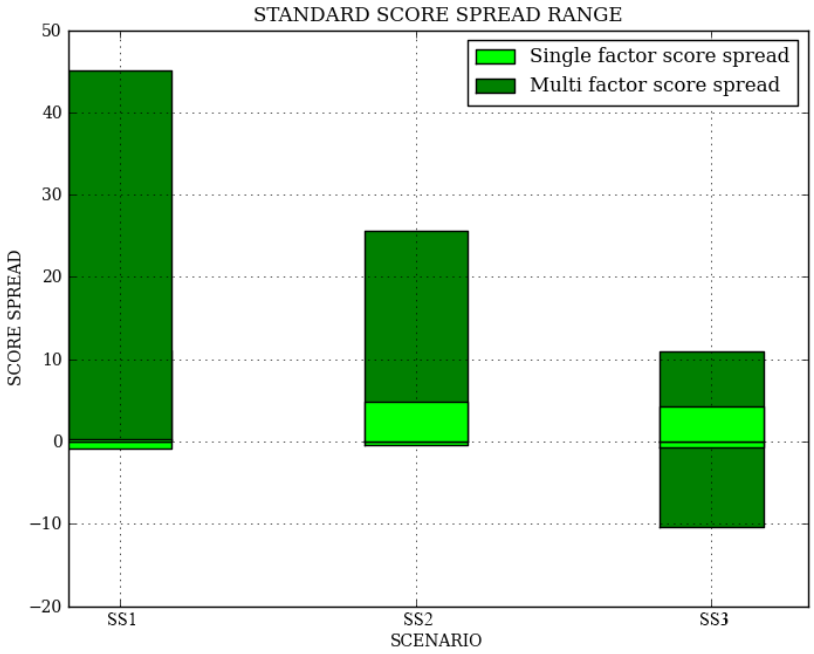
#	Scenario	Single factor score spread	Multi factor score spread
SS1.	Ideal availability & low reliability	[-0.89 , 0.26]	[11.03 , 45.05]
SS2.	Consistent availability & gradually decreasing reliability	[-0.47 , 4.81]	[0.03 , 25.6]
SS3.	Consistent reliability & gradually decreasing availability	[-0.68 , 4.27]	[-10.45 , 10.96]

Table 6.29: Standard score spread summary

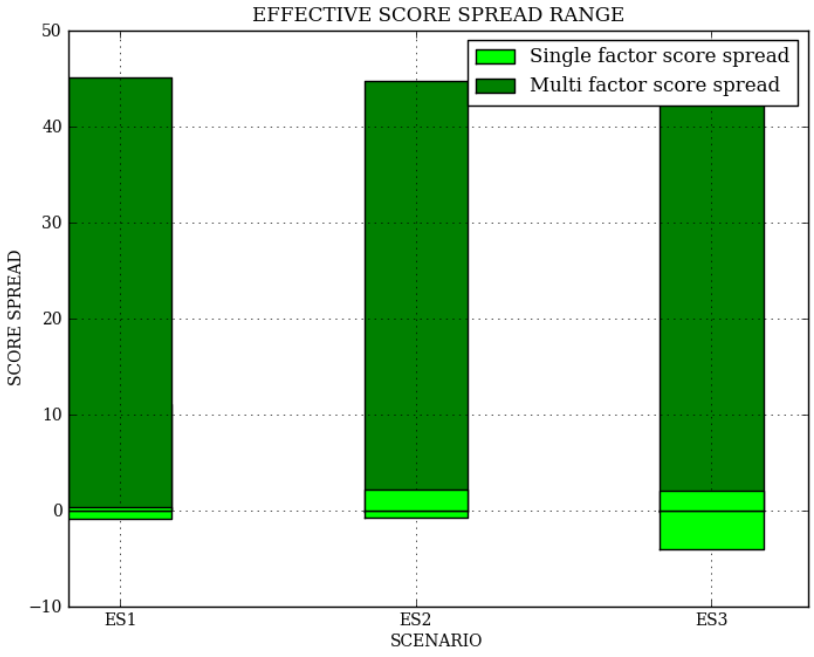
#	Scenario	Single factor effective score spread	Multi factor effective score spread
ES1.	Ideal availability & low reliability	[-0.89 , 0.26]	[11.03 , 45.05]
ES2.	Consistent availability & gradually decreasing reliability	[-0.81 , 2.15]	[-0.14 , 44.67]
ES3.	Consistent reliability & gradually decreasing availability	[-4.05 , 1.99]	[-0.14 , 42.27]

Table 6.30: Effective score spread summary

An additional measurement is intended to reveal the ratio between the number of occasions by which GREPTrust managed to produce a higher score than GridPP to the total number of scores. In order to accommodate this measurement, Figure 6.16 displays two pie charts. The left pie chart breaks down the scores for the single factor batch of experiments (B2). Out of 38 scores, GREPTrust managed to surpass GridPP in only 42% occasions. However, in the multi factor batch of experiments (B3), GREPTrust managed to surpass GridPP in 64% of occasions (69 out of 108 scores). This strengthens the aforementioned discovery regarding GREPTrust capabilities to outperform GridPP in the multi factor batch of experiments.

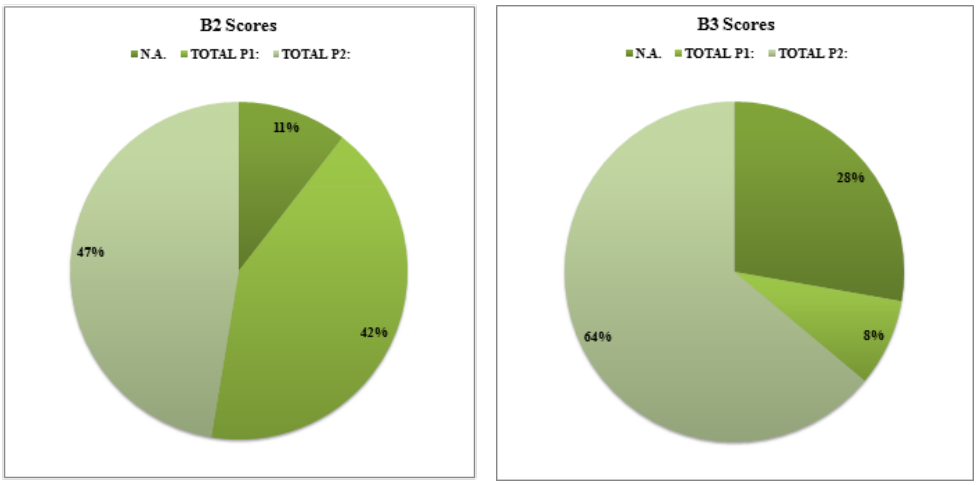


(a) Standard score spread range



(b) Effective score spread range

Figure 6.15: Spread range for standard and effective scores



(a) B2 score distribution range (b) B3 score distribution range

Figure 6.16: Distribution range for batches B2 and B3

6.4 Conclusions

The objective of this chapter was to perform a comparative analysis between GREPTrust and GridPP reputation models based on the developed case study and consequently assess their performance through a series of testbed experiments followed up by a post-mortem analysis. These activities served the purpose of revealing the circumstances by which GREPTrust outperformed GridPP and consequently constituted as an empirical proof for justifying the reputation-policy based trust model. In conclusion, the main outputs of this chapter are the following:

- Series of testbed experiments organised in three different batches: preliminary, single factor and multi factor where each batch contains multiple scenarios.
- Post-mortem analysis which draws conclusions regarding the performance of GREPTrust reputation model in different scenarios based on the testbed experiments and with consideration to the developed financial case study.
- Justification of the reputation-policy based trust model for the scenarios in which GREPTrust outperformed GridPP and a generalisation made via induction regarding the applicability of GREPTrust in computational grids.

Chapter 7

Conclusions

"If we wish to make a new world we have the material ready. The first one, too, was made out of chaos."

Robert Quillen

7.1 Summary

The objective of this research was to introduce a novel reputation paradigm for managing resource selection in Grid computing environments. This was based on the research hypothesis which proclaimed that a synergistic reputation-policy trust model would allow an elevated degree of fine-grained resource selection. The enclosing problem statement included a generalisation of the state of the art and a rationale for considering an exoteric philosophy in trust & reputation management.

Chapter 1 introduced the research framework. It initiated by setting the overall theme, discussing the importance of trust and reputation management in various computing environments and in Grid computing in particular. Additionally, it recognised the need for a robust reputation solution which would optimise resource selection based on client evaluation criteria. This chapter finalised with a description of four research challenges and assumptions as well as an identification of four research contributions which aspired to extend the frontier of current reputation-based trust management systems applied to Grid computing environments.

Chapter 2 provided a thorough background research over the emergence of trust management in various computing environments, the common methodologies for managing trust (i.e. policy-based and reputation-based) and reviewed the key elements of reputation-based trust management systems including common trust evaluation methodologies. A central aspect of this chapter was a discussion of related work regarding reputation-based trust management in Grid computing. It included a literature review spanning the existing solutions leading to a generalisation of the

deficiencies in the state of the art Grid reputation-based trust management systems and made an identification of the main requirements in order to address these deficiencies. The output of this chapter was a solid justification for introducing a novel reputation-policy trust model based on the current deficiencies. This was supported by an hypothetical case study describing two types of analytical jobs which are common in the banking sector and highlighted the key differences between them in terms of QoS requirements. This chapter concluded with a realisation of the need for a solution which would allow external stipulation of QoS requirements based on the given type of job and decisions based on risk aptitude and the global context.

Chapter 3 constituted as a central aspect of this thesis since its aim was to introduce the synergistic reputation-policy based trust model. It made a central proposition of a novel trust paradigm which combines elements of reputation-based and policy-based trust models hence creating a synergy between the two types of trust management methodologies. In contrast to existing esoteric models, the reputation-policy based trust model promoted an exoteric approach, which encouraged an active participation in the trust and reputation evaluation processes; that is a reputation-based trust model in which the metrics for trust and reputation are created, selected and managed externally utilising trust decision strategies which reflect on the submitted job requirements, the global trust context and risk aptitude.

Moreover, the reputation-policy based trust model offered a consolidation of an evaluation model based on multi-aspect discrete trust model and a fuzzy inference based decision model hence tightly coupling the notions of trust and risk and consecutively forming a consolidated fuzzy-discrete trust model. In correspondence with this approach, it aspired to address the deficiencies identified in Chapter 2.

Chapter 4 introduced the Grid Reputation-Policy Based Trust Management Service - GREPTrust. The main objective of the GREPTrust service was to provide a solid reference implementation for the synergistic reputation-policy based trust model. This involved defining a service architecture which incorporated the various artifacts of the reputation-policy based trust model into a single, self-contained Grid reputation-based trust management service. This was performed with consideration to the requirements of the Grid in terms of trust and reputation management including QoS negotiation as well as trust aggregation and Grid resource inference. In addition, Chapter 4 elaborately discussed the reputation querying mechanism provided by the GREPTrust architecture including querying support for a single or a set of resources as well as for aggregated reputation queries and resource inference.

Chapter 5 introduced a dedicated Grid testbed - GREPTrust testbed which enabled the deployment of GREPTrust as a Grid service and stimulate interactions between Grid clients, GREPTrust and Grid resources. In the context of this thesis, this testbed constituted as an infrastructure for an automated testing framework

which streamlined the comparison between exoteric and esoteric reputation-based trust models. A decision was made to compare GREPTrust trust model with the production available GridPP trust model. In addition, this chapter provided an identification of niche area, based on computational finance case study in which GREPTrust was expected to optimise mission critical computations by allowing adaptation for specific job requirements. The chosen comparison job was the volume distribution profile (VDP) generation job which was one of the two jobs described in the hypothetical case study as part of the rationale description in Chapter 2. This chapter finalised with a description of the jobs simulation process and a decision of benchmark criteria for evaluating the performance of Grid jobs based on previously resources recommended by both GridPP and GREPTrust trust models.

Chapter 6 described the experimental results which were based on the case study developed in Chapter 5. This included two sets of experiments: single factor and multi factor. In the single factor experiments both GREPTrust and GridPP trust models solely evaluated the availability factor whereas in the multi factor experiments, GREPTrust was stipulated with both availability and reliability factors whilst GridPP remained with availability. This was conducted in order to demonstrate the advantages of the reputation-policy based trust model in optimising resource selection and as a result increase the quality of the computation. The multi factor experiments clearly demonstrated that an exoteric trust model, in which Grid client applications are endeavoured to stipulate reputation-policy requirements that target specific computing jobs ,outperform esoteric models which rely on single, predefined trust metrics (results demonstrated maximal improvement of about 45%).

7.2 Knowledge Contributions

- The aim of RC1: *Synergistic reputation-policy based trust model* was to introduce an exoteric reputation-based trust model which endeavoured Grid clients (such as resource brokers, schedulers and monitoring toolkits) to gain complete control over the trust and reputation evaluation process with the aim of adaptation of selected resources to specific job requirements. This contribution was achieved by a definition of the trust model structure as well as the data and behaviour of its key artifacts which prompted the metrics for trust and reputation to be managed externally using reputation-policy assertions.

- The aim of RC2: *Grid reputation-policy TMS architecture* was to derive a service oriented based architecture which incorporated the various artifacts of the synergistic reputation-policy based trust model and applied them to a Grid computing environment. This contribution was achieved by a specification of the Grid Reputation-Policy based trust management system (GREPTrust)

providing a solid reference implementation for the synergistic reputation-policy based trust model and its artifacts. This was backed by an elaborative literature review regarding the state of the art Grid reputation-based trust management systems and a study of the affiliated Grid computing requirements.

- The aim of RC3: *Grid resource reputation querying mechanism* was to propose a reputation querying mechanism based on given reputation evaluation criteria. This contribution was achieved by an elaboration on the Query Manager (QM) component of GREPTrust and a design of a querying management mechanism comprising of three steps ranging from transforming a Trust Decision Strategy into a Reputation-Policy Report. Recall the three steps described in Chapter 4, they were comprised of: (i) processing TDS evaluation model, (ii) processing TDS decision model and (iii) generating a reputation-policy report.

- The aim of RC4: *VO aggregated reputation querying mechanism* was to expand on RC3 and extend the reputation querying mechanism to allow trust aggregation based on multiple levels such as a set of resources, organisation and VO. This contribution was achieved by an incorporation of the GREPTrustAggregator (GTA) component which utilised and controlled multiple instances of the Query Manager (QM). Each QM instance computed separately a portion of the evaluation model input dataset (i.e. a subset of the resources or a subset of the participating organisations) thus scaling the GREPTrust solution for very large grids. In addition, RC4 included the Grid resource inference mechanism which allowed inferring about the trustworthiness of a single resource based on the global context (e.g. organisation or VO). This was achieved by introducing the context value parameter to the trust level formula (formula 4.11) and extending the Decision Model (DM) to include an additional fuzzifier element (besides trust value) for the context value input variable (appendix .2.2).

In conclusion, this research has considerably attained its aims and objectives by demonstrating how the synergistic reputation-policy based trust model optimised the resource selection process in computational grids. This was demonstrated in the testbed experiments where GREPTrust managed to utilise computing resources which embraced the characteristics stipulated by the trust decision strategy of the submitted job and consequently managed to mitigate the number of corrupted entries in the volume distribution profile. As a result, the novelty points of a fuzzy-discrete trust model comprising of a discrete evaluation model and a fuzzy decision model as well as the usage of policy assertions to stipulate evaluation requirements established an extension to the state of the art research in reputation management in Grid computing and constituted as an important milestone toward the evolution from failure tolerant research-oriented Grids into mission critical ones.

The strengths of the synergistic reputation-policy based trust model are emerged with its ability to adapt to specific job requirements (such as VDP generation and Monte-Carlo simulation) and provide an optimisation of the reputation evaluation process by basing it on multiple characteristics (e.g. availability, reliability) and contextual factors (e.g. organisation, VO). On a broader scale, since the trust model is designed to be implementation agnostic, it be can be potentially applied to different computing paradigms besides computational grids, such as Web services, REST, XML-RPC and other SOA technologies, which normally involve service consumers and producers negotiating through bilateral agreements. For example, the concept of a trusted agent can be applied to delivered goods and services and not only to computing resources. This can potentially increase the breadth of markets which could make use of the synergistic reputation-policy based trust model as well as the depth of each market - as the model could be used for different types of requests made by service consumers. By contrast, the limitations of the synergistic reputation-policy based trust model include: (i) an increase of communication overhead, as it requires an additional step of stipulating reputation evaluation requirements; (ii) an increase of message complexity, as the size reputation query increases due to the inclusion of the TDS; and (iii) a computational overhead due to the processing of multiple QoS factors and a complex trust decision model.

With regards to the accomplished results, it can be comfortably said that up to a certain degree they were not excessively surprising and that several experimental scenarios were essentially established thus in order to affirm the anticipated behaviour of GridPP and GREPTrust reputation models. This is particularly the case for scenario B3S1 (Ideal availability & low reliability) where GridPP was expected to result in an adverse resource selection and a deterioration of the volume distribution profile as a result. Nevertheless, these tests were conducted with the aim of estimating the projected magnitude by which GREPTrust outperforms GridPP.

Despite the achievements accomplished in this research, it is important to emphasise that the exoteric trust evaluation approach promoted by the synergistic reputation-policy based trust model is suggested as an alternative to esoteric trust models and does not intend to replace them. Esoteric reputation-based trust models offer an adequate level of confidence for most types of users and computational tasks and are generally favoured for their simplicity. On the other hand, the synergistic reputation-policy based trust model targets specific scenarios where explicit stipulation of reputation evaluation criteria is required. From its point of view, it simply attempts to optimise the resource selection process by adapting it to specific type of jobs and user requirements. Nevertheless, it imposes an extra layer of complexity. As a result, both types of reputation modelling approaches (esoteric and exoteric) could coexist, as they target different use case scenarios and business needs.

7.3 Future Work

The work of described in this thesis has also identified a number of areas for future research. These are listed as follows:

1. *Automatic feedback management.* This aspect is currently absent from the reputation-policy based trust model and considered essential . The feedback ratings should be supplied automatically by monitoring toolkits and it should take into account different factors, such as the difference between the actual service provided and the level of service guaranteed in the SLA contract and length of experience the resource broker achieved with the evaluated resource (shorter length implies greater feedback). In addition, the feedback ratings should be supplied for each opinion aspect in the TDS. For example, if the resource broker stipulated availability and reliability as quality factors, it should supply ratings for these two aspects on transaction completion.

2. *Fortification of the Reputation-Policy Trust Model.* Several improvements and additions can be made to the trust model. For example, at current state the, only historical rating feedbacks are considered as valid trust source for evaluation. The model can be expanded with real time performance data (via monitoring toolkits) as well as trust prediction models which would aim to assess the performance of a resource at a certain point in the future. This functionality can be proven useful for future scheduled jobs in which solely relying on historical information would not be sufficient. In addition, improvements for VO Aggregated reputation querying mechanism could optionally introduce a VO level TDS which weighs the importance of its underlying organisations. This would improve on the current state of the model which uses standard mean to calculate the aggregated value of a group of resources.

3. *Automatic TDS selection/generation.* At the current state, the creation, selection and management of relevant trust decision strategies is the exclusive responsibility of the trusting agents. It would be highly captivating to incorporate an automatic TDS selection/generation utility which would either assign or generate a TDS file based on the desired job reputation requirements and weigh it against the intended execution capacity, the depth of feedback ratings, the global trust context and general risk aptitude. This could be based on linear programming for analysing the different constraints and sophisticated data mining techniques that recognise complex patterns and make intelligent decisions based on these patterns. This feature would require cross discipline efforts between Grid computing and machine learning algorithms.

Glossary

AM Admin Manager. 84, 118

ARR Average of Reputation Ratings. 35, 42

AT Algorithmic Trading. A utilisation 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. 120

availability Proportion of time a system is in a functioning condition. 50, 53, 62, 64, 74, 116, 121, 135

bin A time interval container which stores the average percent of stock volume traded during that interval. 52, 119

CA Certification Authority. 39

CCP Child Correlation Process. A child process spawned by the correlation process. Each child process operates on a single opinion matrix generating a trust value for that opinion. 74, 89, 90

CD Context Data. 66, 72, 74, 86, 89, 90

CE Computing Element. A cluster of Grid resources that carries out the execution of a job. 116, 121

CMP Customised Matrices Pool. 59, 65, 68, 74–76, 86, 90, 92, 94

context the nature of a service or service functions (e.g. 'store data') or alternatively defined as an object or entity (e.g. 'file system'). 19, 65, 66

context value An aggregated trust value of an organisation or VO. 70, 101, 103, 104, 108, 109

COT Cut-off time. 22, 74

CP A matchmaking process which involves a reconciliation of each opinion element stipulated in the trust decision strategy with it's aggregated historical ratings in each opinion matrix counterpart thus in order to compute trust values. 23, 63–65, 72, 74, 75, 86

direct trust Trust relationship which is based on the personal experience a trusting agent has achieved with a trusted agent. 32

DM Decision Model. 70, 72, 84, 86, 96, 98, 99, 108, 109, 116, 135, 140, 142, 144–146, 149, 150, 153, 156–159, 161, 162, 172

DMV Depth Mean Value. 106

DR Decision Rule. An if-then statement used for mapping trust and context values to trust levels. 71

DRE Decision Rule Engine. A rule engine acting as a trustworthiness scale for producing trust levels from trust values. 63–65, 72, 76, 84, 86–88, 96, 99, 100

DTT Direct Trust Table. 35

effective score the ratio between the number of valid and missing entries and the total number of entries in a VDP. 131, 140, 142, 143, 154, 160, 165, 166

ELC Execution Load Capacity. The ratio between the number of utilised resources and the total number of resources in the VO. 139, 140, 142, 144–146, 149, 150, 153, 156–159, 161, 162

EM Evaluation Model. 67, 68, 70, 72, 74, 76, 86, 89, 140, 142, 144–146, 149, 150, 153, 156–159, 161, 162

execution risk An adverse effect on a running job as a result of a failure of one or more Grid resources. 18

execution trend collection of score spreads. 140, 143, 146, 152, 154, 160, 164

FCR Freedom of Choice for Resources. 42

feedback value The actual value which expresses the feedback for a given opinion. 61, 62, 85, 113

FIS Fuzzy Inference System. 87, 96

FM Feedback Manager. 84

FRS Fuzzy Rule Set. 96

GIS Grid Information System. 104

GREPTrust Grid Reputation-Policy Based Trust management system. 22, 25, 79–86, 102, 106–108, 110, 112, 122, 134–136, 139, 141–144, 146, 149, 150, 152–155, 160, 164, 165, 168, 170–172

GridPP Grid for Particle Physics. 35, 42, 49, 50, 111, 112, 114–116, 118, 122–124, 126, 129, 131–135, 139, 141, 142, 144, 146, 149, 150, 152, 154, 155, 160, 164, 165, 168, 171

GSI Grid Security Infrastructure. 39

GTA GREPTrustAggregator. 102, 172

indirect trust Trust relationship in which trust is evaluated from ratings provided by other trusting agents regarding their personal experiences with the trusted agent. 32

IVS Instant Virtual System. 47

MDS Monitoring and Discovery Service. 26

multi-faceted An important characteristic of reputation which means that reputation is comprised of multiple aspects such as availability reliability honesty and etc. 19, 21, 23, 48, 51, 56, 57, 59, 62, 67, 78

OA Opinion Accumulator. 89

OM Opinion matrices. Tabular data structures which store the historical evaluation feedback values reported by trusting agents. 23, 68, 74–76

opinion An opinion is a general impression of a recommendation agent regarding a trusted agent derived from its feedbacks on all the transactions that were conducted with the trusted agent. In the context of this thesis an opinion refers to an impression regarding a single quality aspect (i.e. availability reliability and etc). 18, 56, 59, 61, 63, 68, 69, 74, 75, 85

OSM Opinion Summary Matrix. A specialised matrix which stores the separate opinion trust values for each evaluated trusted agent. 74

policy a statement of the intent of the owner or controller of some computing resources specifying how he wants them to be used. 19

QM Querying Manager. 86, 118, 172

QoP Quality of Protection. 48

QoS Quality of Service. The fulfilment of the service agreement or mutually agreed service. 18, 28, 33, 40, 45, 48, 50, 51, 53, 56, 58, 63, 66, 68, 70, 78, 80–82

R Rule. 78

RA Reputation Algorithm. 64–66, 72, 86, 87, 89, 94, 96, 100

RC Resource Consumer. 30

recommendation See reputation feedback. 20

reliability The ability of a system or component to perform its required functions under stated conditions for a specified period of time. 48, 50, 53, 62, 64, 74, 116, 135

reputation An aggregation of all the recommendations from the third-party recommendation agents regarding the service quality of the trusted agent in a given context and time slot. 18

reputation feedback A statement issued by a recommendation agent regarding the quality of service provided by a trusted agent in a single transaction. Each reputation feedback is comprised of an opinion and feedback value. 18, 20, 22, 31–33, 57–59, 61–64, 74, 75, 84, 85

reputation query An inquiry made by a trusting agent for a specific context and time slot regarding a service product or a trusted agent. The reputation query typically includes a context ID context description context time and etc. 19, 20, 22, 65, 66

reputation value An aggregated value that represents the total recommendations made by third-party recommendations agents regarding the trustworthiness of the trusted agent. 19, 32

reputation-policy assertions A set of reputation evaluation requirements. 20, 22, 56

RI Report Item. 77

RP Resource Provider. 30

RPDS Reputation Policy Data Store. 84, 118

RPQ Reputation-Policy Query. A reputation query fortified with reputation-policy assertions for stipulating reputation evaluation criteria. 20, 64, 65, 72, 86

RPR Reputation-Policy Report. 26, 65, 72, 77, 84, 86, 87, 100

RTF Recommender Trust Factor. A mathematical formula which assigns lower weight for business allies recommendations when evaluating the trustworthiness of a resource. 43, 59

RTT Reputation Trust Table. 35

RVT Resource Value Table. A hashtable comprising of resource identifiers and trust values. 74, 86, 89

SADS SEDOL Analytics Data Store. 129

SAM Service Availability Monitoring. 42, 112, 116

SE Storage Element. A Grid resource which stores the data required or generated by the computing element. 116, 121

SEDOL Stock Exchange Daily Official List. A list of security identifiers used in the United Kingdom and Ireland for clearing purposes. 120

SLA Service Level Agreement. 23, 40, 48, 51, 56, 57, 62, 80, 81

spread The difference between the results of two execution scores (e.g. standard scores - P2SS-P1SS or effective scores - P2ES-P1ES. 135, 140, 143, 146, 152, 154, 160, 165, 166

SRM Storage Resource Manager. 116

SSO Single Sign On. 39

standard score the ratio between the number of valid entries and the total number of entries in a VDP. 131, 140, 142, 143, 154, 160, 165

symbol Short abbreviation used to uniquely identify publicly traded shares of a particular stock on a particular stock market. For example the symbol ‘MKS‘ is an abbreviation for Marks & Spencer. 119, 120

TCR Trust Comparison Report. A structured trust evaluation report produced by the GREPTrust testbed. The report contains each resource ID and the trust levels assigned to it by GridPP and GREPTrust. A comparison report serves as input to the volume distribution profile job simulation. 111, 112, 128, 133

TDF Trust Decay Function. A mathematical formula aggregated to a trust function which assigns older ratings less weight and therefore decreases their importance when calculating trust values. 22, 43, 57, 74

TDS See trust decision strategy. 23, 26, 60, 64, 66, 72, 74, 84, 86, 96, 109, 112, 113, 122, 135, 172, 174

TDS repository Storage facility for trust decision strategy documents. 59, 60, 64, 83, 118

term Specifies a membership function with a distinct name (e.g. 'poor' 'good' 'excellent') and a series of axis points defining the length and scale of the membership function. 70

third-party recommendation agent An entity which provides a reputation feedback regarding the quality of a service product or a trusted agent. 18, 20, 32, 36, 58, 61, 62, 74, 75

TMS Trust Management System. A a system which is responsible for managing trust in a distributed environment. 29

trust A subjective belief a trusting agent has in the capability and willingness of a trusted agent to deliver a quality service or product in a given context and time slot. 17, 63

trust decision strategy A structured document stipulating reputation evaluation criteria. It is comprised of an Evaluation Model (EM) as well as a Decision Model (DM). 21, 23, 56, 63, 86, 89

trust function A mathematical formula for computing trust values. 29, 32–36

trust level A value produced by a trustworthiness scale that determines the amount of trust that a trusting agent has in a trusted agent. The trust level is measured against the trustworthiness scale. 18, 21, 22, 24, 56, 58, 59, 63, 64, 66, 71, 81, 84, 87, 96, 98–101, 108, 109, 112, 172

trust management An activity of collecting encoding analysing and presenting evidence relating to competence honesty security or dependability with the purpose of making assessments and decisions regarding trust relationships. 28, 50, 51

trust metrics Refers to the actual value that expresses the trust (or trustworthiness) of an entity provided by the trust model. 29, 30, 33, 34, 42, 46, 47

trust model A quantitative model for producing trust values (i.e. scores) from the interactions of agents or from the recommendations or history of agents. This includes esoteric trust models and exoteric trust models. 23, 30, 31, 33

trust relationship A bond or association between a trusting agent and a trusted agent. 18

trust source An input value to a trust model such as direct or experience or a combination of direct and experience. 33, 69

trust value A numerical representation of a trust relationship between a trusting agent and a trusted agent. 18, 22, 31–33, 56, 57, 63, 64, 66, 67, 70, 74, 86, 87, 90–94, 96, 97, 99–101, 104, 106–109, 172

trusted agent An entity in whom faith has been placed by another entity in a given context and at a given time slot. 18, 61–64, 66, 67

trusting agent An entity who has faith or belief in another entity in a given context and at a given time slot. 17, 58–64, 66, 72, 74, 76

trustworthiness A measure or an estimate of the level or the degree of trust utilising a well-defined trustworthiness scale. 15, 24, 48, 57, 61, 63

trustworthiness scale A system which provides a metric that helps determining the trust level that a trusting agent should have in a trusted agent. 18, 20, 21, 76

TS Terms Set. A set comprising of a group of terms. 70

utility function A mathematical function which measures of the relative satisfaction from consumption of a product or service. 44, 62

VDP See Volume distribution profile. 26, 52, 53, 111, 119, 126, 128, 135, 136, 139, 165, 171

VO Virtual Organisation. 24, 26, 42, 44–46, 50, 60, 61, 66, 68, 79–81, 83–85, 101, 102, 104, 108, 110, 113

VOMS Virtual Organisation Membership Service. 39

VWAP Volume Weighted Average Price. Trading benchmark calculated as the ratio of the value traded to total volume traded over a particular time horizon. It is a measure of the average price a stock traded at over the trading horizon. 119

VWAP trading algorithm Schedule driven trading strategy which processes orders over specified time horizon and spreads each trade in proportion to the generated historical volume distribution. 119, 131, 133

Bibliography

[1] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The Condor experience. In *Concurrency and Computation: Practice and Experience*, 2004.

[2] D.H.J. Epema, M. Livny., R. van Dantzig, X. Evers & Pruyne, J., A Worldwide flock of Condors: Load Sharing amon Workstation Clusters. *Journal on Future Generations of Computer Systems*, vol. 12(1):53-56, Elsevier, 1996.

[3] J. P. Jones, PBS: portable batch system, In *Beowulf cluster computing with Linux table of contents*, Cambridge, MA: MIT Press, pp. 369-390, 2001.

[4] M.Q. Xu, Effective Metacomputing Using LSF Multicluster, In *Proceedings of the First International Symposium*, Cluster Computing and the Grid (CCGrid '01), pp. 100-105, 2001.

[5] J. Frey, Condor DAGMan: Handling Inter-Job Dependencies, Technical report, University of Wisconsin, Dept. of Computer Science, 2002.

[6] Rajasekar, A., Wan M., Moore, R., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhu, B., Chen, S.-Y. & Olschanowsky , R., Storage resource broker - managing distributed data in a grid. *Computer Society of India Journal, Special Issue on SAN*, 33(4), pp. 42-54, October 2003.

[7] I. Foster, C. Kesselman The Grid: Blueprint for a new computing infrastructure. Morgan Kaufman, pp. 259-278, 1999.

[8] I. Foster, C. Kesselman, S. Tuecke The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High-Performance Computing Applications, pp. 200-222, 15(3), 2001.

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

[10] Chang, E., Dillon, T.S., Hussain, F., Trust and reputation for service oriented environments- Technologies for building business intelligence and consumer confidence, John Wiley & Sons, ISBN: 0-470-01547-0, 2005.

[11] S.Marsh. Formalizing Trust as a Computational Concept. PhD thesis. Scotland, University of Stirling, 1994.

[12] L. Dimitriou. Financial services Grid virtualization for increased business performance and lower TCO. In *Grid in Finance*, NY, 2006.

[13] D. Gambetta. Trust: Making and Breaking Cooperative Relations, Chapter Can We Trust Trust?, Department of Sociology, University of Oxford, pp. 213-237, 1988.

[14] F. Azzedin and M. Maheswaran. Towards Trust-Aware Resource Management in Grid Computing Systems. In *CCGRID '02: Proc. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp.452-457. IEEE Computer Society, 2002.

[15] F. Azzedin and M. Maheswaran. Integrating trust into grid resource management systems. In *Proc. 2002 International Conference on Parallel Processing (ICPP'02)*, pp. 47-54. IEEE Computer Society, 2002.

[16] F. Azzedin and M. Maheswaran. Evolving and Managing Trust in Grid Computing Systems. In *Proc. of IEEE Canadian Conference on Electrical & Computer Engineering*, pp. 1424-1429. IEEE Computer Society, 2002.

[17] F. Azzedin and M. Maheswaran. A Trust Brokering System and its Application to Resource Management in Public-Resource grids. In *Proc. 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*. IEEE Computer Society, 2004.

[18] Mui, L., Mohtashemi M. and Halberstadt A. A Computational Model of Trust and Reputation, Available: http://www.cnn.com/2002/WORLD/europe/10/04/world.cities/, 2002.

[19] Mui, L., Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks, PhD thesis, Massachusetts Institute of Technology, 2002.

[20] Wang, Y. and Vassileva, J., Baysian network-based trust model, in *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Melbourne, Australia, 2003.

[21] T. Grandison and M. Sloman. A Survey of Trust in Internet Applications. IEEE Communications Surveys and Tutorials, 3(4), September 2000.

[22] A. Jøsang, R. Ismail and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618-644, March 2007.

[23] A. Jøsang, Trust and reputation systems, in *Foundations of Security Analysis and Design IV*, FOSAD 2006/2007 - Tutorial Lectures, (Bertinoro, Italy), Springer LNCS 4677, September 2007.

[24] A. Jøsang, S. Lo Presti, Analysing the relationship between risk and trust, in: T. Dimitrakos (Ed.), In *Proceedings of the Second International Conference on Trust Management*, Oxford, March, 2004.

[25] P. Resnick and R. Zeckhauser. Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. In M.R. Baye, editor, The Economics of the Internet and E-Commerce, vol. 11 of Advances in Applied Microeconomics. Elsevier Science, 2002.

[26] B. Yu and M.P Singh. A Social Mechanism of Reputation Management in Electronic Communities. In *Proceedings of the 4th International Workshop on Cooperative Information Agents*, pp. 154-165, 2000.

[27] A. Singh and Ling Liu. Trustme: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pp. 142–149, IEEE Computer Society, Washington D.C., USA, 2003.

[28] G. von Laszewski, B.E. Alunkal, and I. Veljkovic. Towards reputable grids. Scalable Computing: Practice and Experience, 6(3):95–106, 2005.

[29] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, May 2003.

[30] B. Alunkal. Grid EigenTrust: A Framework for Computing Reputation in Grids. MSc thesis, Department of Computer Science, Illinois Institute of Technology, 2003.

[31] A.Abdul-Rahman and S.Hailes. Supporting Trust in Virtual Communities. In *Proceedings of the Hawaii International Conference on Systems Sciences* 33, 2000.

[32] S. Naqvi, P. Massonet, A. Arenas. A Study of Languages for the Specification of Grid Security Policies, Technical Report TR-0037, CoreGrid, April 2006.

[33] I. Dionysiou, H. Gjermundrod, D. E. Bakken. An Initial Approach for Adaptive Trust in Grid Environments, In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, Computation World*, pp. 719-722, November 2009.

[34] Amazon, Web page. Available online at: http://www.amazon.com.

[35] eBay, Web page. Available online at: http://www.ebay.com.

[36] iTunes, Web page. Available online at: http://www.itunes.com.

[37] M. Richardson, R. Agrawal, P. Domingos, Trust Management for the Semantic Web, In *Proceedings of the 2nd International Semantic Web Conference*, pp. 351-368, 2003.

[38] P. Bonatti, C. Duma, D. Olmedilla, *et. al.*, An Integration of Reputation-based and Policy-based Trust Management. In *Proceedings of the Semantic Web Policy Workshop*, Galway, Ireland, November 2005.

[39] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Proceedings of the 7th ACM Conference on Computer and Communication Security*, 2000.

[40] P. Bonatti and D. Olmedilla. Driving and Monitoring Provisional Trust Negotiation with Metapolicies. In *IEEE 6th International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pp. 14-23, Stockholm, Sweden, IEEE Computer Society, June 2005.

[41] R.Gavriloaie, W.Nejdl, D.Olmedilla, *et. al.*, No Registration Needed: How to Use Declarative Policies and Negoatiation to Access Sensetive Resources on the Semantic Web. In *1st European Semantic Web Symposium (ESWS 2004)*, pp. 342-356, Heraklion, Crete, Greece, May 2004.

[42] N. Li and J. Mitchell. RT: A Role-based Trust-management Framework. In *DARPA Information Survivability Conference and Exposition (DISCEX)*, Washington, D.C., April 2003.

[43] K.Aberer and Z.Despotovic, Managing trust in a peer-2-peer information system. In *Proceedings of 10th International Conference on Information and Knowledge Management*, 2001.

[44] L. Kagal, Rei: A Policy Language for the Me-Centric Project, HP Labs Technical Report, HPL-2002-270, 2002.

[45] N. Damianou, N. Dulay, E. Lupu, M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, Imperial College, UK, Research Report Department of Computing, 2001.

[46] M. Johnson, P. Chang, R. Jeffers, J. Bradshaw et al. KAoS Semantic Policy and Domain Services: An Application of DAML to Web Services-Based Grid Architectures, Proceedings of the AAMAS 03 workshop on Web Services and Agent-Based Engineering, Melbourne, Australia, July 2003.

[47] A. S. Vedamuthu, D. Orchad, F. Hirsh, M. Hondo, P. Yendluri, T. Boubez, U. Yalcinalp, WS-Policy Specification, http://www.w3.org/Submission/WS-Policy/, Web Services Policy 1.5 - Framework, September 2007.

[48] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. Communications of ACM, 43(12):45–48, December 2000.

[49] A. Arenas, M. Wilson, and B. Matthews, On Trust Management in Grids, in *Autonomics '07: Proceedings of the 1st international conference on Autonomic computing and communication systems*. ICST, Brussels, Belgium, Belgium:ICST (Institute for Computer Sciences, Social-Informaticsand Telecommunications Engineering), pp. 1–7, 2007.

[50] CoreGrid. D.ia.03 survey material on trust and security. Technical Report D.IA.03, CoreGrid, October 2005.

[51] M. Adamski, G. Jankowski, N. Meyer, A. Arenas, B. Matthews, M. Wilson, A., P. Fragopoulou, V. Georgiev, A. Hevia, and J. Platte. Trust and Security in Grids: A State of the Art, May 2008.

[52] G.C. Silaghi, A.E. Arenas, Luis M. Silva, Reputation-based trust management systems and their applicability to grids. CoreGRID. Technical Report TR-0064, *Institutes on Knowledge and Data Management & System Architecture, CoreGRID Network of Excellence*, February 2006.

[53] G. C. Silaghi, A.E. Arenas, L.M. Silva. A utility-based reputation model for service-oriented computing. Toward Next Generation Grids, in *Proceedings of the CoreGRID Symposium 2007*. Springer, 2007.

[54] A. E. Arenas, B. Aziz, G. C. Silaghi, Reputation Management in Grid-Based Virtual Organisations In *Proc. International Conference on Security and Cryptography (SECRYPT 2008)*, Porto, Portugal, 26-29 Jul 2008.

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

[56] K. Czajkowski, I. Foster, C. Kesselman. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, LNCS 2537, pp. 153-183, Springer 2002.

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

[58] F. Kerschbaum, J. Haller, Y. Karabulut, and P. Robinson. Pathtrust: A trust-based reputation service for virtual organization formation. In *iTrust2006, 4th International Conference on Trust Management*, vol. 3986, Lecture Notes in Computer Science, pp. 193-205. Springer, 2006.

[59] J. Jia and X. Qu, Towards Trustworthy Resource Selection in Grid: A Fuzzy Partial Ordering based Approach, International Journal of Computer Science and Network Security, vol. 6 No.9A, September 2006.

[60] S. Song, K. Hwang and M. Macwan, Fuzzy Trust Integreation for Security Enforcement in Grid Computing. NPC 2004, LNCS 3222, 2004.

[61] S. Song, K. Hwang, Y.K. Kwok. Trusted Grid Computing with Security Binding and Trust Integration. Journal of Grid Computing, vol. 3, no. 1, pp. 24-34, 2005.

[62] L. A. Zadeh, Fuzzy Sets, Information and Control 8, pp. 338-353, 1965.

[63] L. A. Zadeh. Fuzzy Control, Fuzzy Graphs, and Fuzzy Inference, Future Directions of Fuzzy Theory and Systems, eds. Y. Yam and K. S. Leung, (World Scientific, Singapore, 1995) 1-9, 1995.

[64] Timothy J. Ross. Fuzzy Logic with Engineering Applications. McGraw-Hill, New York, 1995.

[65] Q. Zhang, T. Yu, K. Irwin. A Classification Scheme for Trust Functions in Reputation-Based Trust Management. Workshop on Trust, Security, and Reputation on the Semantic Web, Hiroshima, Japan, 2004.

[66] M. Brinkløv, R. Sharp. Incremental Trust in Grid Computing. ccgrid, pp. 135-144, Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), 2007.

[67] J. Sonnek and J. Weissman, A Quantitative Comparison of Reputation Systems in the Grid, in *Proceedings of the Sixth ACM/IEEE International Workshop on Grid Computing*, 2005.

[68] S. Song and K. Hwang, Trusted Grid Computing with Security Assurance and Resource Optimization. In *Proc. 17th International Conference on Parallel and Distributed Computing Systems (PDCS-2004)*, September 2004.

[69] W. Nejdl, D. Olmedilla, M. Winslett. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In *Proc. of the Workshop on Secure Data Management in a Connected World (SDM'04)*, Springer, Toronto, Canada, 2004.

[70] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pp. 144–152, ACM Press, New York, NY, USA, 2003.

[71] S. Buchegger and J.Y. Le Boudec. Self-Policing Mobile Ad-Hoc Networks by Reputation. IEEE Communication Magazine, 43(7):101–107, 2005.

[72] Y. Rebahi, V. Mujica, and D. Sisalem. A Reputation-Based Trust Mechanism for Ad hoc Networks. In *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC'05)*, pp. 37–42, IEEE Computer Society, Washington D.C., USA, 2005.

[73] E. Mamdani and S. Assilian. An Experiment Linguistic Synthesis With a Fuzzy Logic Controller, In *International Journal of Man-Machine Studies*, vol. 7, No. 1, pp. 1-13, 1975.

[74] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, *IEEE Trans. Systems Man Cybernet*, vol. 15, SMC-IS, no. 1, pp. 116-131, 1985.

[75] S. Song and K. Hwang and M. Macwan, Fuzzy Trust Integration for Security Enforcement in Grid Computing, In *Network and Parallel Computing*, vol. 3222 of *Lecture Notes in Computer Science*, pp. 9-21. Springer Verlag, October 2004.

[76] R. Buyya and M. Murshed, GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, In *J. Concurrency and Computation: Practic-Experience (CCPE)*, vol. 14, pp. 1175–1220, 2002.

[77] Y. Zetuny, G. Terstyanszky, S. Winter and P. Kacsuk Reputation-Policy Trust Model for Grid Resource Selection In *DAPSYS2008, 7th International Conference on distributed and parallel systems*. Debrecen, Hungary. September 3-5, 2008.

[78] Y. Zetuny, G. Terstyanszky, S. Winter, P. Kacsuk Articulating Subjective Trust-Based Decision Strategies Utilizing the Reputation-Policy Trust Management Service Conf. Proc. of the UK e-Science 2008 All Hands Meeting, Edinburgh, September 10-13, 2008.

[79] Y. Zetuny, G. Terstyanszky, S. Winter and P. Kacsuk Adapted Quality Resource Selection Using the Grid Reputation-Policy Trust Management Service Conf. Proc. of the IADIS International Conference Informatics 2009, Algarve, Portugal, ISBN: 9789728924867, June 17-23 2009.

[80] A. Chakrabarti Managing Trust in the Grid, Grid Computing Security pp. 221-222, ISBN: 9783540444923, Springer, 2007.

[81] D. Olmedilla, O. F. Rana, B. Matthews, W. Nejdl. Security and Trust Issues. In *Semantic Grids. Proc. Schloss Dagstuhl Seminar No. 05271: Semantic Grid: The Convergence of Technologies*, 04-08 July 2005.

[82] IEC 1131 - Programmable Controllers Part 7 - Fuzzy Control Programming International Electrotechnical Commission (IEC), Technical Commitee No. 65: Industrial Process Measurement and Control Sub-Committee 65 b: Devices, January 1997.

[83] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artif. Intell. Rev. 24 (1), pp. 33–60, 2005.

[84] J. Sabater, C. Sierra. Regret: a reputation model for gregarious societies, In *Fourth Workshop on Deception, Fraud and Trust in Agent Societies.* ACM Press, 2001.

[85] Y. Li, M. Mascagni. Grid-Based Monte Carlo Applications, Lecture Notes in Computer Science, 2536: 13-24, Proceedings of the International Workshop/-Conference on Grid Computing, GRID2002, Baltimore, 2002.

[86] K. Gor, R.A. Dheepak, S. Ali, L.D. Alves, N. Arurkar, I. Gupta, A. Chakrabarti, A. Sharma, S. Sengupta. Scalable Enterprise Level Workflow and Infrastructure Management in a Grid Computing Environment. In *Proc. Cluster Computing and Grid (CCGrid)*, Cardiff (Wales), pp. 661-667, 2005.

[87] S. Andreozzi, D. Antoniades, A. Ciuffoletti, A. Ghiselli, E.P. Markatos, M. Polychronakis, P. Trimintzios. Issues about the Integration of Passive and Active Monitoring for Grid Networks. In *Proc. CoreGrid Integration Workshop*, Pisa, Italy, 2005.

[88] B. Coghlan, A. Djaoui, S. Fisher, Magowan, M. Oevers. Time, information services and the grid. In Advances in Database Systems (BNCOD): *Supplement to the Proceedings of the 18th British National Conference on Databases at RAL*, pp. 9–11, 2001.

[89] J.M. Schopf, M.D. Arcy, N. Miller, L. Pearlman, I. Foster, C. Kasselman. Monitoring and Discovery in Web Services Framework: Functionalities and Performance of Globus Toolkit's MDS4. Tech. Report, ANL/MCS-P1248-0405, 2005.

[90] A. Cooke, W. Nutt, J. Magowan, P. Taylor, J. Leake, R. Byrom, L. Field, S. Hicks, M. Soni, A. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B. Coghlan, S. Kenny, D. O'Callaghan, and J. Ryan. Relational Grid Monitoring Architecture (R-GMA), In *Proceedings of UK e-Science All Hands Meeting*, pp. 744–750, 2003.

[91] F. Hussain, E. Chang, and O. Hussain. State of the art review of the existing bayesian-network based approaches to trust and reputation computation. In *2nd International Conference on Internet Monitoring and Protection*, pp. 154 – 158, 2007.

[92] Yao Wang, Julita Vassileva, Trust and Reputation Model in Peer-to-Peer Networks, In *Proceedings of the 3rd IEEE International Conference on Peer-to-Peer Computing*, pp. 150-158, Linköping, Sweden; 1-3 September 2003.

[93] Yao Wang, Julita Vassileva, Bayesian Network-Based Trust Model, In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence (WI2003)*, pp. 372-378, Halifax, Canada, 13-16 October 2003.

[94] T. Grandison. *Trust Management for Internet Applications.* PhD thesis, Imperial College London, July 2003.

[95] M. Blaze, J. Feigenbaum, and J.Lacy. Decentralized trust management. In *Proceedings of IEEE Conference on Security and Privacy*, 1996.

[96] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the 3rd European Symposium on Research in Computer Security.* Springer-Verlag, 1994.

[97] G. Caronni. Walking the web of trust. In *Proceedings of 9th IEEE International Workshops on Enabling Technologies (WETICE)*, pp. 153–158, June 2000.

[98] K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Proceedings of Ninth International Conference on Cooperative Information Systems*, 2001.

[99] C. Duma, N. Shahmehri, and G. Caronni. Dynamic trust metrics for peer-to-peer systems. In *Proceedings of 2nd IEEE Workshop on P2P Data Management*, Security and Trust, August 2005.

[100] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. Eigenrep: Reputation management in p2p networks. In *Proceedings of 12th International WWW Conference*, pp. 640–651, 2003.

[101] I. Domowitz and H. Yegerman. The Cost of Algorithmic Trading A First Look at Comparative Performance. Edited by Brian R. Bruce, Algorithmic Trading: Precision, Control, Execution. Institutional Investor, 2005.

[102] Yang, J.; Jiu, B.: Algorithm Selection: A Quantitative Approach. In *Algorithmic Trading II - Precision, Control, Execution*, pp. 26–34, 2006.

[103] N. Li and J. C. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages* (PADL 2003), pp. 58–73, 2003.

[104] A. Ghosh, A. Chakrabarti, R.A. Dheepak, S. Ali S, I. Gupta. Streamlining Drug Discovery Research by Leveraging Grid Workflow Manager. In *Life Sciences Grid (LSGrid)*, Singapore, 2005.

[105] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.

[106] R. Alfieri et al. VOMS: An Authorization System for Virtual Organizations. In *Proceedings of 1st European Across Grids Conference*, Santiago de Compostela, 2003.

[107] J. Basney, M. Humphrey, V.Welch. The MyProxy Online Credential Repository. IEEE Software Practice and Experience, vol. 35, issue 9, pp. 801-816, 2005.

[108] S. Boeyen et al. Liberty Trust Models Guidelines. In *J. Linn (editor)*, Liberty Alliance Project. Liberty Alliance, version 1.0, 2003.

[109] S. Cantor. Shibboleth Architecture. Available online at: http://shibboleth.internet2.edu/, Version 5. 11, May 2002.

[110] L. A. Cornwall, J. Jensen et al. Authentication and Authorization Mechanisms for Multi-Domain Grid Environments. *Journal of Grid Computing*, 2(4). pp. 301-311. 2004.

[111] Chadwick, D.W. and Otenko, A., The PERMIS X.509 Role Based Privilege Management Infrastructure. 7th ACM Symposium on Access Control Models and Technologies, 2002.

[112] Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K. and Essiari, A., Certificate-based Access Control for Widely Distributed Resources. 8th Usenix Security Symposium, 1999.

[113] A.E. Arenas, I. Djordjevic, T. Dimitrakos, L. Titkov, J. Claessens, C. Geuer-Pollman, E.C. Lupu, N. Tuptuk, S. Wesner, L. Schubert. Towards Web Services Profiles for Trust and Security in Virtual Organisations. IFIP Working Conference on Virtual Enterprises - PRO-VE'05, Valencia, Spain, 2005.

[114] S. Wesner, L. Schubert, T. Dimitrakos. Dynamic Virtual Organizations in Engineering. In *Proceedings of German-Russian Workshop*, 2005.

[115] P.R. Zimmermann. The Official PGP User's Guide. MIT Press, 1995.

[116] J. Woodcock and J. Davies. Using Z: Specification, Refinement, and Proof. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[117] EU FP6 GridTrust Project. Available online at: http://www.gridtrust.eu/gridtrust/

[118] P. Thibodeau. Wachovia uses grid technology to speed up transaction apps. Available online at: http://www.computerworld.com/s/article/9000476/Wachovia-uses-grid-technology-to-speed-up-transaction-apps, 2006.

[119] IBM Banking: Grid Technology Helps ZKB Perform Efficient Financial Modelling Calculations. Available online at: http://www.scribd.com/doc/22669635/IBM-Banking-Grid-Technology-Helps-ZKB-Perform-Efficient-Financial-Modelling-Calculations, 2009.

[120] Success Story - Deutsche Bank's Securities Custody Reporting System. Available online at: http://www.boic.org/case-d.htm

[121] Enabling Grids for E-sciencE (EGEE) Project. Available online at: http://www.eu-egee.org

[122] LHC Computing Grid Technical Site. Available online at: http://lcg.web.cern.ch/LCG/

[123] The Globus Alliance Project. Available online at: http://www.globus.org

[124] China National Grid (CNGrid) Project. Available online at: http://i.cs.hku.hk/ clwang/grid/CNGrid.html

[125] The National Grid Service (NGS) Project. Available online at: http://www.grid-support.ac.uk

[126] TeraGrid. Available online at: http://www.teragrid.org

[127] Globus Toolkit. Available online at: http://www.globus.org/toolkit/

[128] gLite. Available online at: http://glite.web.cern.ch/glite/

[129] OASIS XACML specification. Available online at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[130] Van Inwagen, Peter, An Essay on Free Will, Oxford: Clarendon Press, 1983.

[131] JFuzzyLogic. Available online at: http://jfuzzylogic.sourceforge.net/html/index.html

[132] JPype bridge library. Available online at: http://jpype.sourceforge.net/

[133] GridPP Project. Available online at: http://www.gridpp.ac.uk/

[134] D. Britton et al, GridPP: the UK grid for particle physics , UK e-Science All Hands Conference, Phil. Trans. R. Soc. A, June 28, 2009.

[135] UK SAM Test Results http://pprc.qmul.ac.uk/ lloyd/gridpp/samtest.html

[136] SAS Grid Computing Project. Available online at: http://www.sas.com/technologies/architecture/grid

[137] Oracle Grid Computing Project. Available online at: http://www.oracle.com/technologies/grid/index.html

[138] IBM Grid computing Project. Available online at: http://www-03.ibm.com/grid

[139] Sun Grid Engine Project. Available online at: http://www.sun.com/software/sge

[140] National Weather Service (NWS). Available online at: http://nws.noaa.gov/

[141] European Organization for Nuclear Research (CERN). Available online at: http://www.cern.org

[142] NGS P-GRADE portal, Web page. Available online at: http://www.cpc.wmin.ac.uk/cpcsite/index.php

Appendices

.1 Dissemination of Research Findings

.1.1 Publications

Y. Zetuny, G. Kecskemeti, G. Terstyanszky, S. Winter, T. Kiss and P. Kacsuk (2005) Automatic Deployment and Interoperability of Grid Services. In: Cox, Simon J. and Walker, D.W., (eds.) Proceedings UK e-Science All Hands Meeting 2005, Steering via the Image in Local, Distributed and Collaborative Settings. EPSRC, Swindon, UK, pp. 729-736.

Y. Zetuny, G. Terstyanszky, S. Winter and K. Madani (2005) Grid Services Interoperability Within the Context of a Workflow. In: microCAD 2005 International Scientific Conference. microCAD, Miskolc, Hungary, pp. 403-408.

Y. Zetuny and G. Kecskemeti, T. Kiss, G. Sipos, P. Kacsuk, G. Terstyanszky and S. Winter (2005) Deployment and interoperability of legacy code services. In: Gorlatch, Sergei and Danelutto, Marco, (eds.) Proceedings of the CoreGRID Integration Workshop, Pisa, Italy. Universita di Pisa, Dipartimento di Informatica, Pisa, Italy, pp. 377-386.

Y. Zetuny, G. Terstyanszky, S. Winter and P. Kacsuk Reputation-Policy Trust Model for Grid Resource Selection In DAPSYS2008, 7th International Conference on distributed and parallel systems. Debrecen, Hungary. September 3-5, 2008.

Y. Zetuny, G. Terstyanszky, S. Winter and P. Kacsuk Articulating Subjective Trust-Based Decision Strategies Utilizing the Reputation-Policy Trust Management Service Conf. Proc. of the UK e-Science 2008 All Hands Meeting, Edinburgh, September 10-13, 2008.

Y. Zetuny, G. Terstyanszky, S. Winter and P. Kacsuk Adapted Quality Resource Selection Using the Grid Reputation-Policy Trust Management Service Conf. Proc. of the IADIS International Conference Informatics 2009, Algarve, Portugal, June 17-23 2009.

.1.2 Awards

Best Research Student Submission and Presentation, ECS Research Conference, University of Westminster, June 25-26 2009.

PhD Students Award 2010 (runner-up position) ECS Research Conference, University of Westminster, July 1-2 2010.

.2 GREPTrust Diagrams, Sample Files & Outputs

.2.1 Diagrams

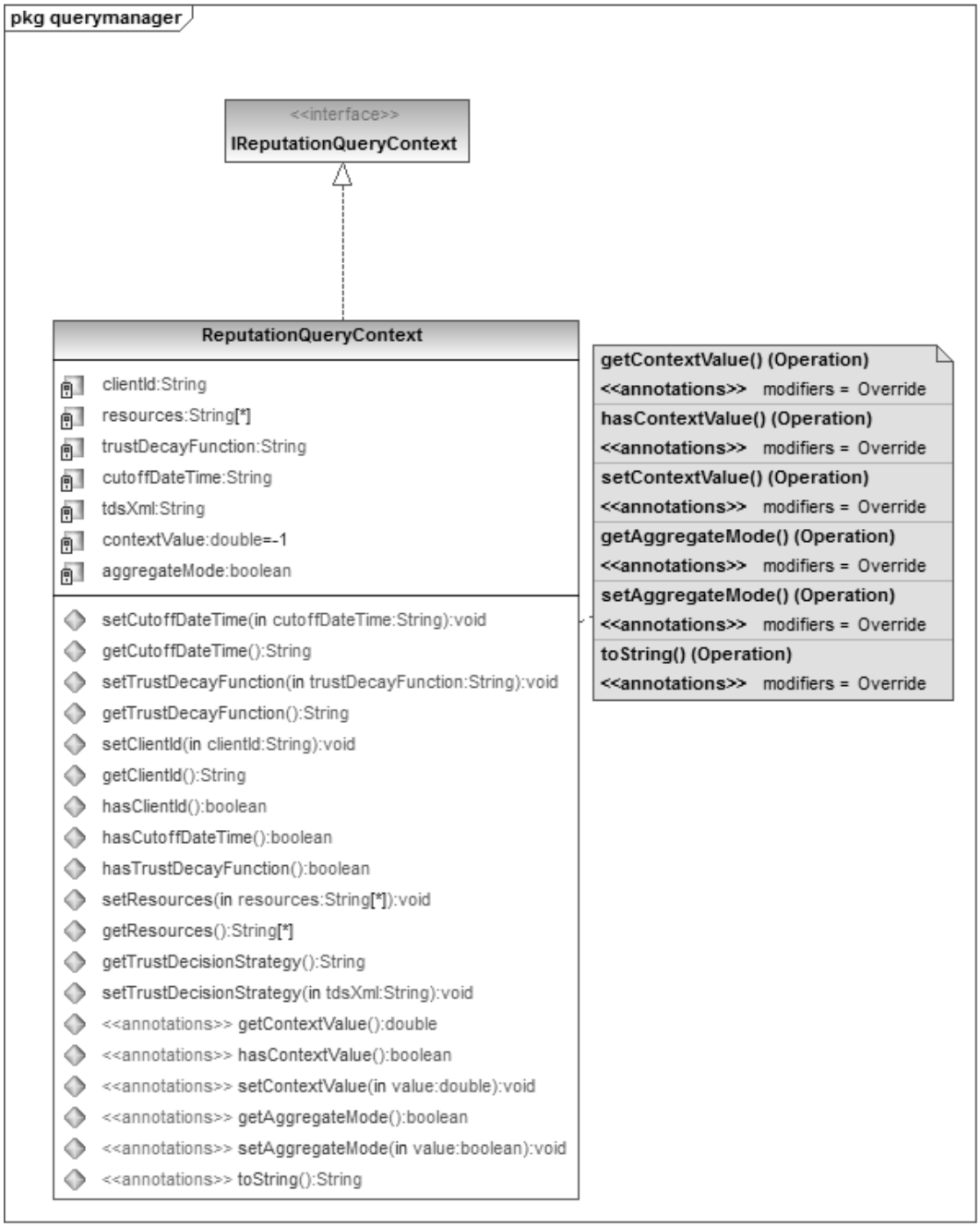


Figure 1: Reputation-Policy Query Class Diagram

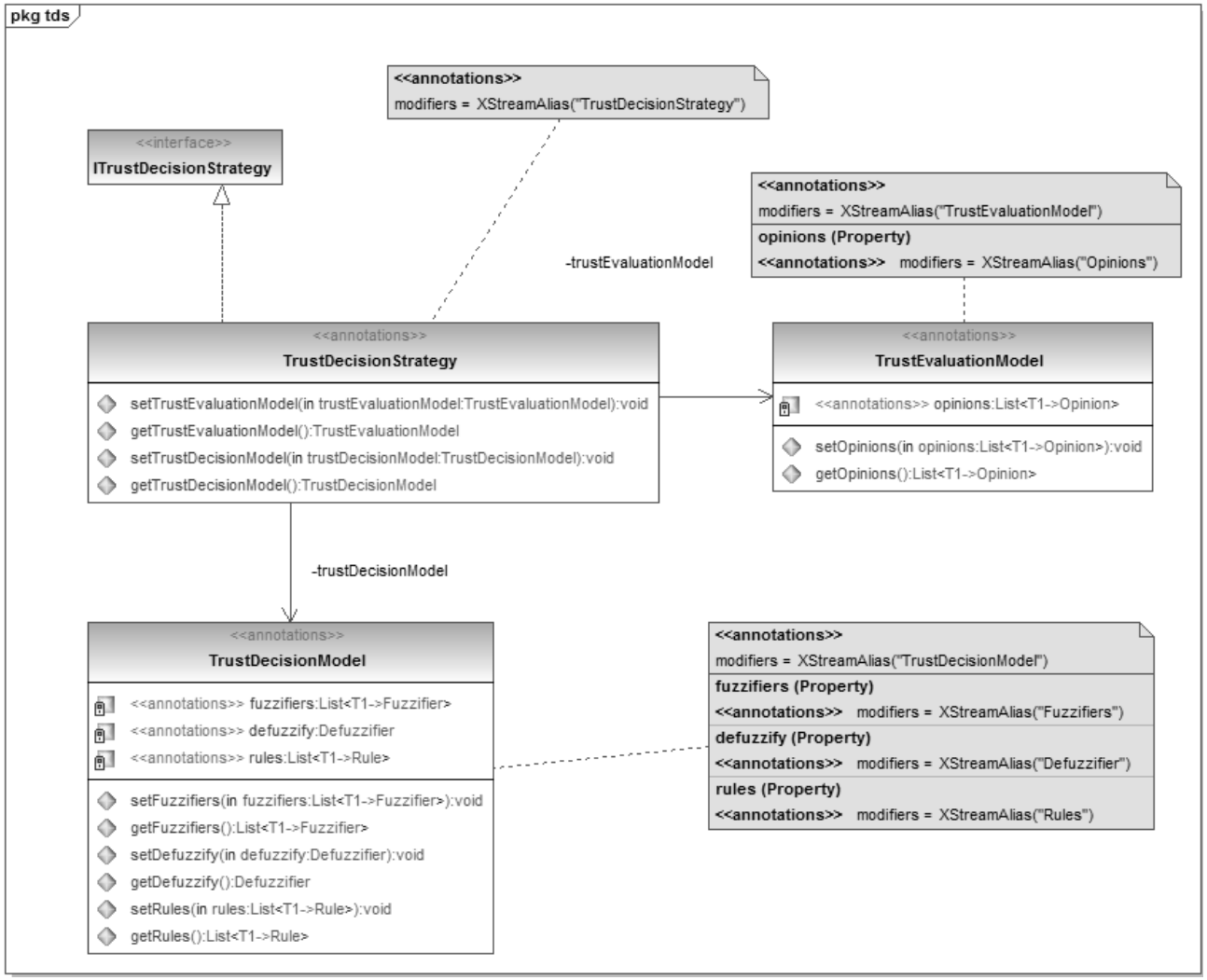


Figure 2: Trust Decision Strategy Class Diagram

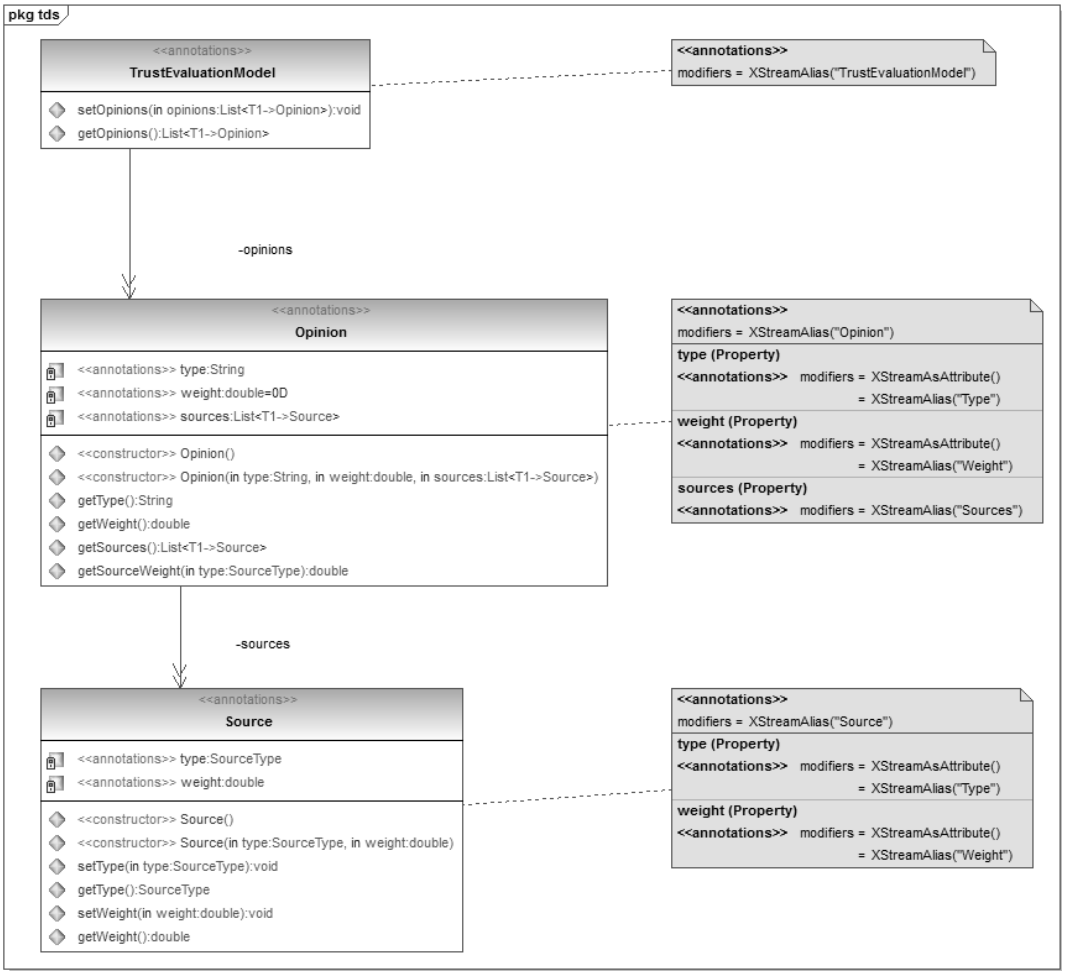


Figure 3: Evaluation Model Class Diagram

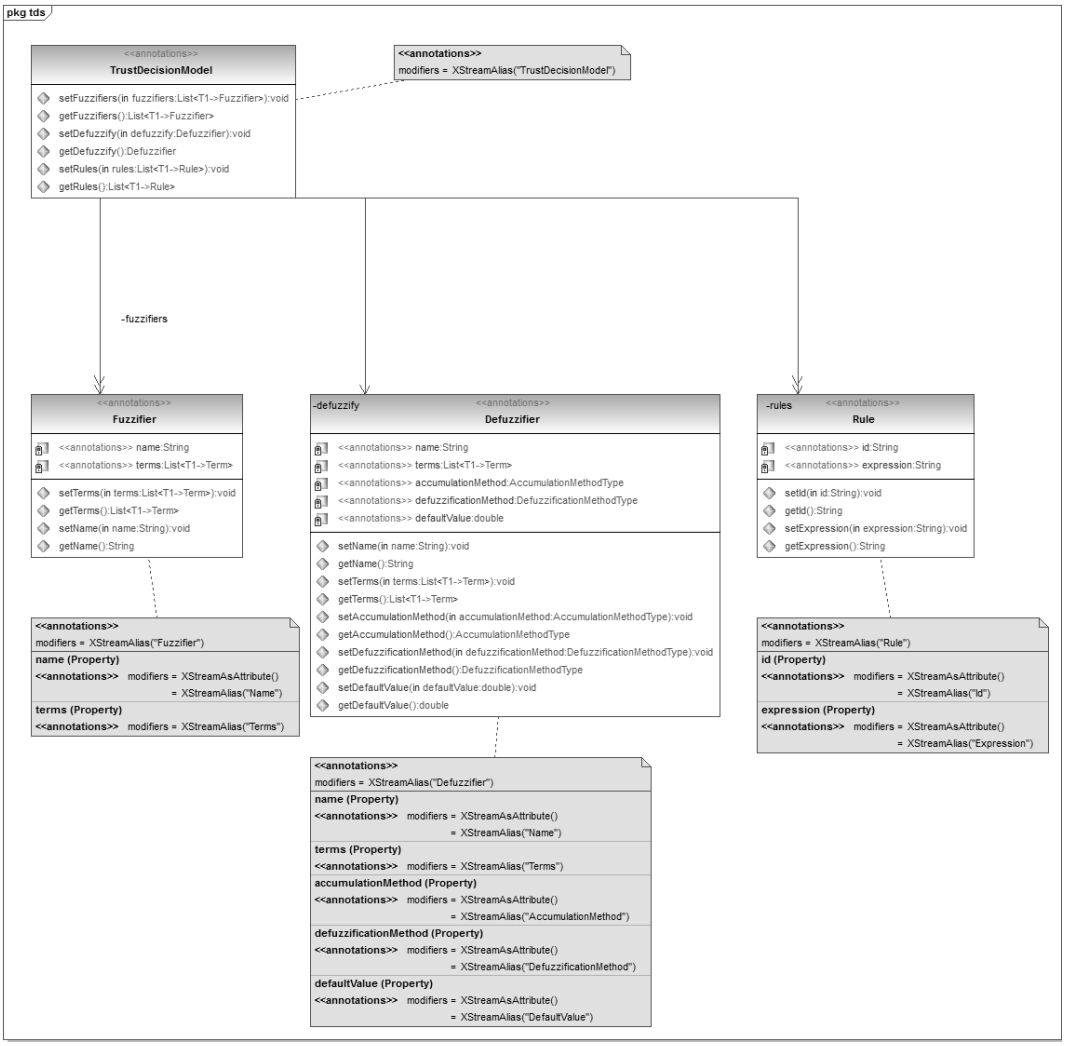


Figure 4: Decision Model Class Diagram

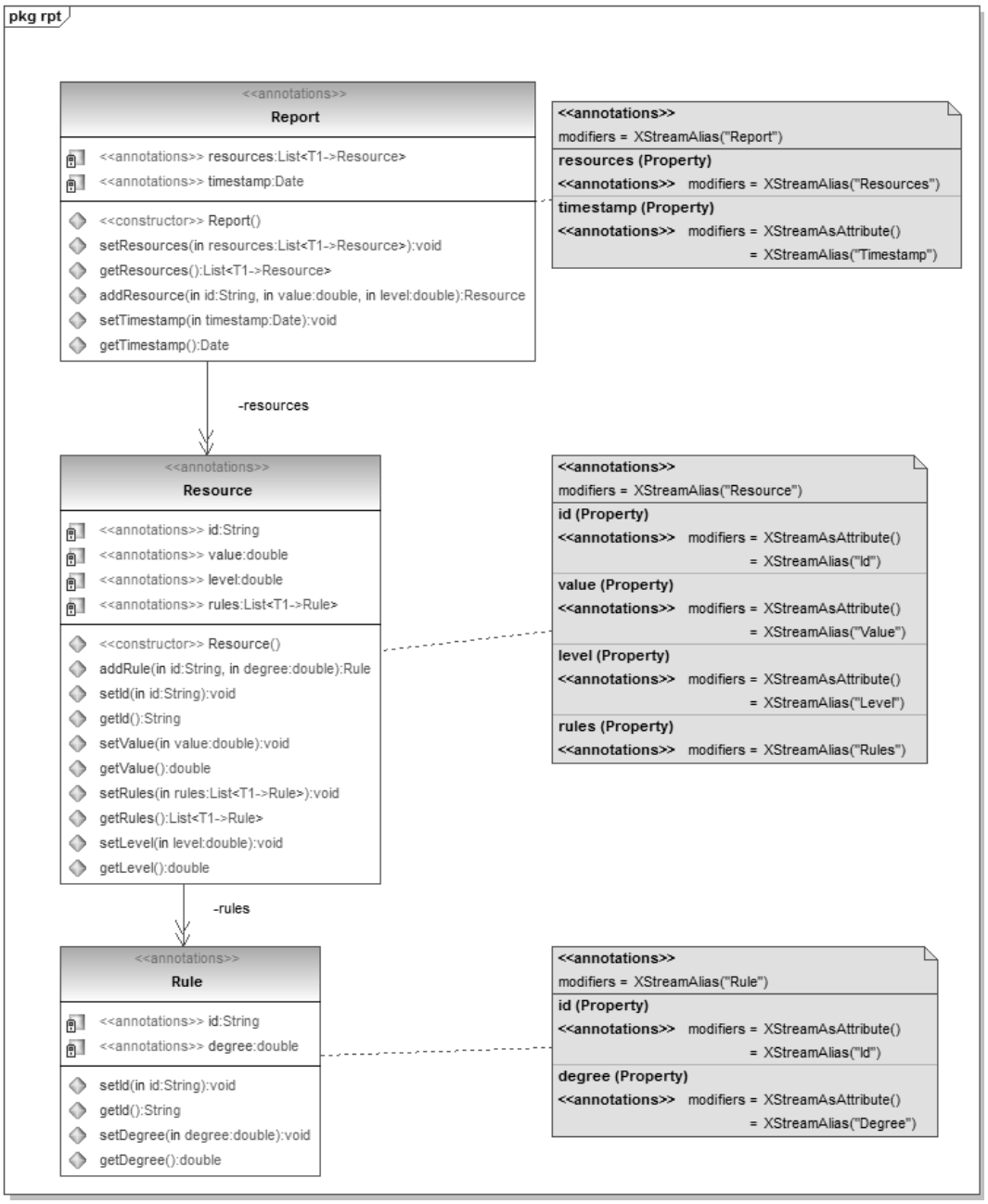


Figure 5: Reputation-Policy Report Class Diagram

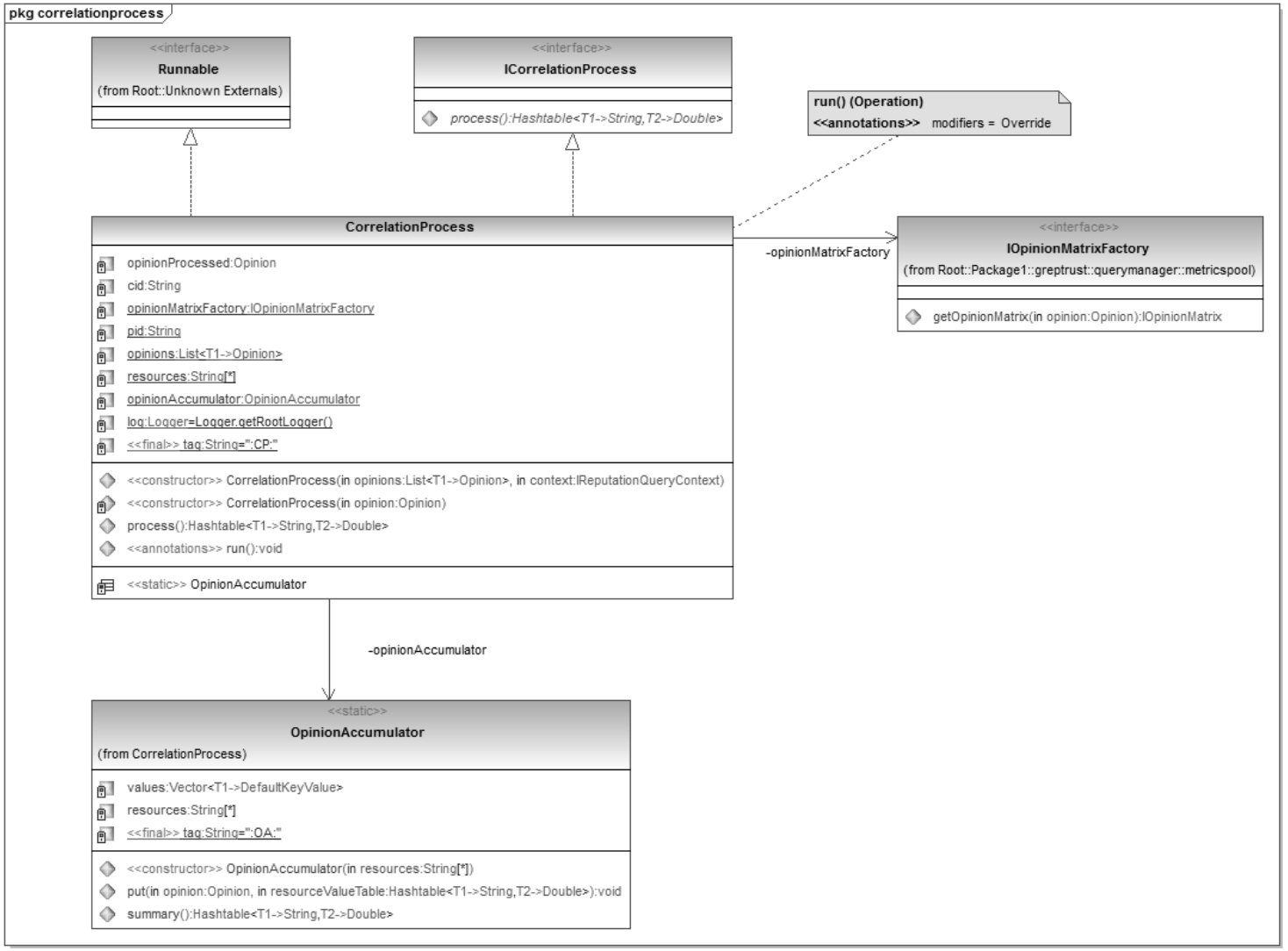


Figure 6: Correlation Process Class Diagram

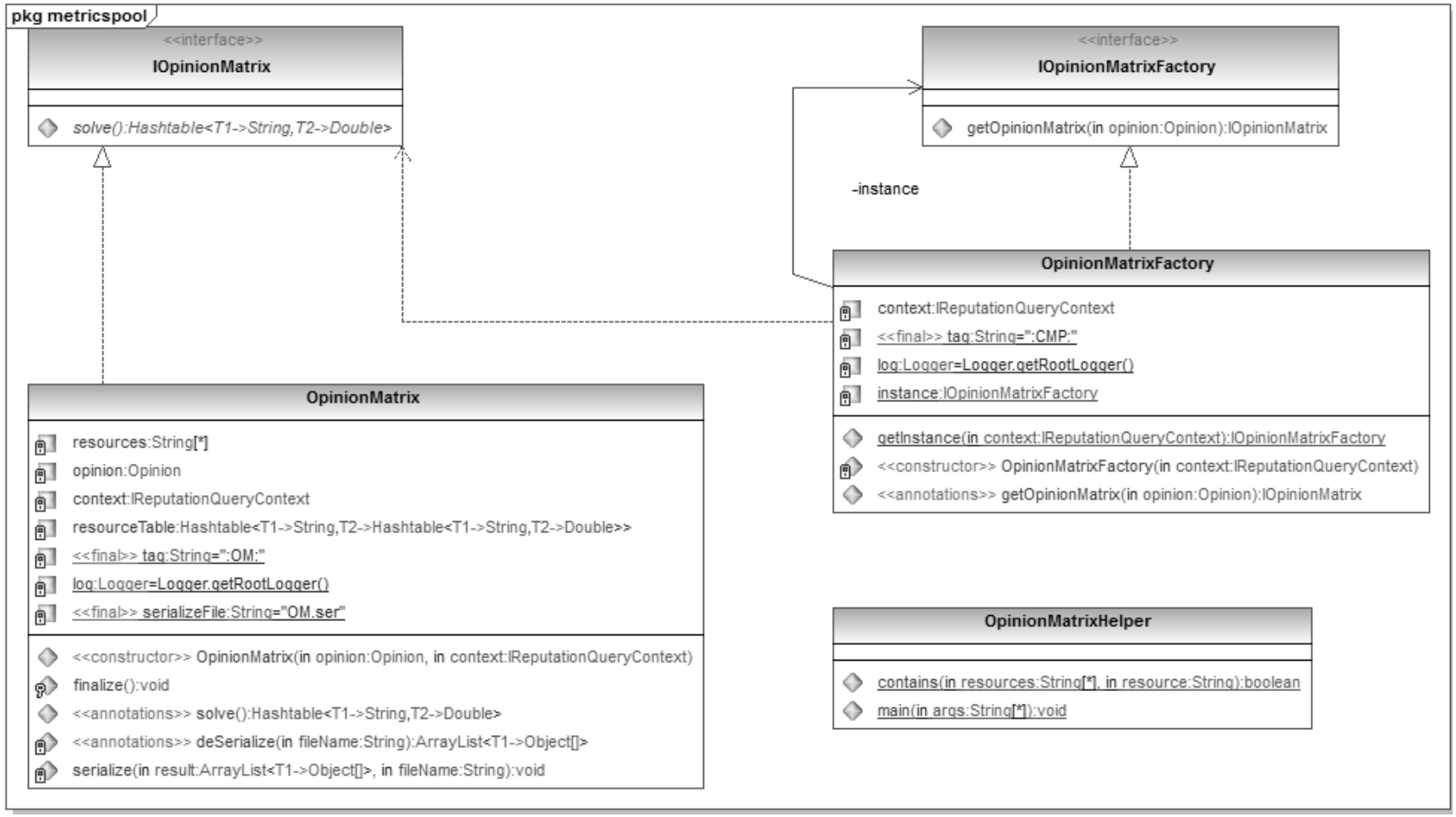


Figure 7: Customised Matrices Pool Class Diagram

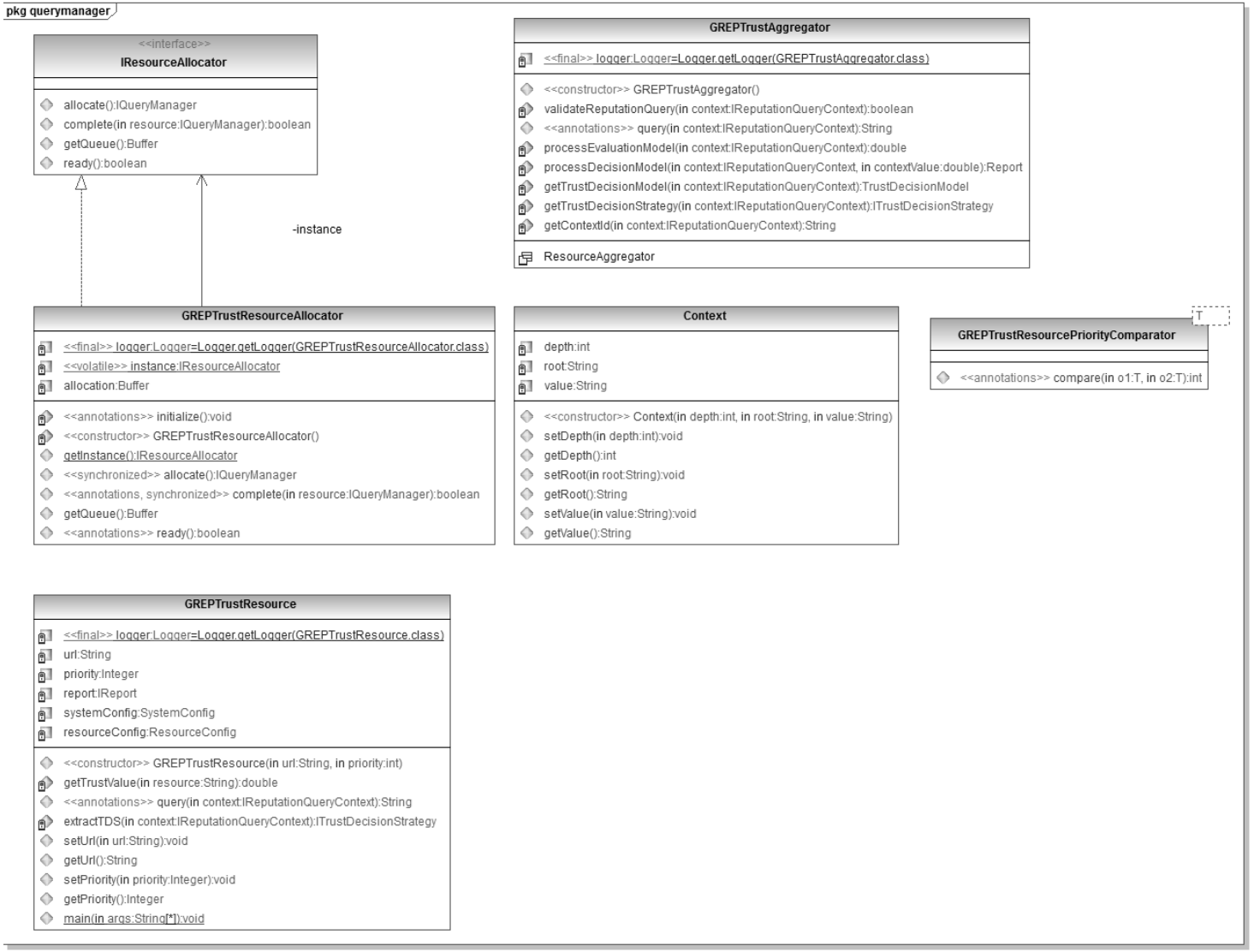


Figure 8: GREPTrustAggregator Class Diagram

.2.2 Sample Files

Sample TDS File (1)

```
<TrustDecisionStrategy>
  <TrustEvaluationModel>
    <Opinions>
      <Opinion Type="1" Weight="1">
        <Sources>
          <Source Type="Experience" Weight="0.5"/>
          <Source Type="Reputation" Weight="0.5"/>
        </Sources>
      </Opinion>
    </Opinions>
  </TrustEvaluationModel>
  <TrustDecisionModel>
  <Fuzzifiers>
    <Fuzzifier Name="trust_value">
      <Terms>
        <Term Name="poor">
        <Points>
          <Point X="0.0" Y="1.0" />
          <Point X="0.2" Y="0.0" />
        </Points>
        </Term>
        <Term Name="good">
        <Points>
          <Point X="0.2" Y="0.0" />
          <Point X="0.4" Y="1.0" />
          <Point X="0.6" Y="0.0" />
        </Points>
        </Term>
        <Term Name="excellent">
        <Points>
          <Point X="0.58" Y="0.0" />
          <Point X="1.0" Y="1.0" />
        </Points>
        </Term>
      </Terms>
    </Fuzzifier>
    </Fuzzifiers>
    <Defuzzifier Name="trust_level" AccumulationMethod="MAX"
      DefuzzificationMethod="COG" DefaultValue="0">
      <Terms>
        <Term Name="none">
```

```
42          <Points>
43            <Point X="0.0" Y="0.0" />
44            <Point X="0.1" Y="1.0" />
45            <Point X="0.2" Y="0.0" />
46          </Points>
47          </Term>
48          <Term Name="limited">
49          <Points>
50            <Point X="0.2" Y="0.0" />
51            <Point X="0.5" Y="1.0" />
52            <Point X="0.8" Y="0.0" />
53          </Points>
54          </Term>
55          <Term Name="full">
56          <Points>
57            <Point X="0.8" Y="0.0" />
58            <Point X="1.0" Y="1.0" />
59          </Points>
60          </Term>
61        </Terms>
62      </Defuzzifier>
63      <Rules>
64      <Rule Id="1" Expression="IF trust_value IS poor
65 THEN trust_level IS none" />
66      <Rule Id="2" Expression="IF trust_value IS good
67 THEN trust_level IS limited" />
68      <Rule Id="3" Expression="IF trust_value IS excellent
69 THEN trust_level IS full" />
70    </Rules>
71    </TrustDecisionModel>
72 </TrustDecisionStrategy>
```

Sample TDS File (2)

```
1  <TrustDecisionStrategy>
2    <TrustEvaluationModel>
3      <Opinions>
4        <Opinion Type="1" Weight="0.9">
5          <Sources>
6            <Source Type="Experience" Weight="0.9"/>
7            <Source Type="Reputation" Weight="0.1"/>
8          </Sources>
9        </Opinion>
10       <Opinion Type="2" Weight="0.1">
11         <Sources>
```

```
12          <Source Type="Experience" Weight="0.9"/>
13          <Source Type="Reputation" Weight="0.1"/>
14        </Sources>
15      </Opinion>
16    </Opinions>
17  </TrustEvaluationModel>
18  <TrustDecisionModel>
19    <Fuzzifiers>
20      <Fuzzifier Name="trust_value">
21        <Terms>
22          <Term Name="poor">
23          <Points>
24            <Point X="0.0" Y="1.0" />
25            <Point X="0.5" Y="0.0" />
26          </Points>
27          </Term>
28          <Term Name="good">
29          <Points>
30            <Point X="0.0" Y="0.0" />
31            <Point X="0.5" Y="1.0" />
32            <Point X="1.0" Y="0.0" />
33          </Points>
34          </Term>
35          <Term Name="excellent">
36          <Points>
37            <Point X="0.5" Y="0.0" />
38            <Point X="1.0" Y="1.0" />
39          </Points>
40          </Term>
41        </Terms>
42      </Fuzzifier>
43      <Fuzzifier Name="context_value">
44        <Terms>
45          <Term Name="poor">
46          <Points>
47            <Point X="0.0" Y="1.0" />
48            <Point X="0.5" Y="0.0" />
49          </Points>
50          </Term>
51          <Term Name="good">
52          <Points>
53            <Point X="0.4" Y="0.0" />
54            <Point X="0.6" Y="1.0" />
55            <Point X="0.8" Y="0.0" />
```

```
56          </Points>
57          </Term>
58          <Term Name="excellent">
59          <Points>
60            <Point X="0.7" Y="0.0" />
61            <Point X="1.0" Y="1.0" />
62          </Points>
63          </Term>
64        </Terms>
65      </Fuzzifier>
66    </Fuzzifiers>
67    <Defuzzifier Name="trust_level" AccumulationMethod="MAX"
        DefuzzificationMethod="COG" DefaultValue="0">
68      <Terms>
69        <Term Name="none">
70        <Points>
71          <Point X="0.0" Y="0.0" />
72          <Point X="0.1" Y="1.0" />
73          <Point X="0.2" Y="0.0" />
74        </Points>
75        </Term>
76        <Term Name="limited">
77        <Points>
78          <Point X="0.2" Y="0.0" />
79          <Point X="0.5" Y="1.0" />
80          <Point X="0.8" Y="0.0" />
81        </Points>
82        </Term>
83        <Term Name="full">
84        <Points>
85          <Point X="0.8" Y="0.0" />
86          <Point X="0.9" Y="1.0" />
87          <Point X="1.0" Y="0.0" />
88        </Points>
89        </Term>
90      </Terms>
91    </Defuzzifier>
92    <Rules>
93      <Rule Id="1" Expression="IF␣trust_value␣IS␣poor␣and␣
          context_value␣is␣poor␣THEN␣trust_level␣IS␣none" />
94      <Rule Id="2" Expression="IF␣trust_value␣IS␣good␣and␣
          context_value␣is␣poor␣THEN␣trust_level␣IS␣limited" />
95      <Rule Id="3" Expression="IF␣trust_value␣IS␣excellent␣and␣
          context_value␣is␣poor␣THEN␣trust_level␣IS␣limited" />
```

```
96      <Rule Id="4" Expression="IF trust_value IS poor and
            context_value is good THEN trust_level IS none" />
97      <Rule Id="5" Expression="IF trust_value IS good and
            context_value is good THEN trust_level IS limited" />
98      <Rule Id="6" Expression="IF trust_value IS excellent and
            context_value is good THEN trust_level IS full" />
99      <Rule Id="7" Expression="IF trust_value IS poor and
            context_value is excellent THEN trust_level IS limited" /
            >
100     <Rule Id="8" Expression="IF trust_value IS good and
            context_value is excellent THEN trust_level IS full" />
101     <Rule Id="9" Expression="IF trust_value IS excellent and
            context_value is excellent THEN trust_level IS full" />
102   </Rules>
103 </TrustDecisionModel>
104 </TrustDecisionStrategy>
```

Sample RPR File

```
1  <Report Timestamp="2009-10-26 20:06:07.734 GMT">
2    <Resources>
3      <Resource Id="19" Value="0.42" Level="0.63">
4        <Rules>
5          <Rule Id="3" Degree="0.17"/>
6          <Rule Id="2" Degree="0.0"/>
7          <Rule Id="1" Degree="0.0"/>
8        </Rules>
9      </Resource>
10     <Resource Id="18" Value="0.49" Level="0.65">
11       <Rules>
12         <Rule Id="3" Degree="0.27"/>
13         <Rule Id="2" Degree="0.0"/>
14         <Rule Id="1" Degree="0.0"/>
15       </Rules>
16     </Resource>
17     <Resource Id="17" Value="0.48" Level="0.65">
18       <Rules>
19         <Rule Id="3" Degree="0.26"/>
20         <Rule Id="2" Degree="0.0"/>
21         <Rule Id="1" Degree="0.0"/>
22       </Rules>
23     </Resource>
24     <Resource Id="16" Value="0.48" Level="0.65">
25       <Rules>
26         <Rule Id="3" Degree="0.26"/>
```

```
27          <Rule Id="2" Degree="0.0"/>
28          <Rule Id="1" Degree="0.0"/>
29        </Rules>
30      </Resource>
31      <Resource Id="15" Value="0.38" Level="0.62">
32        <Rules>
33          <Rule Id="3" Degree="0.11"/>
34          <Rule Id="2" Degree="0.0"/>
35          <Rule Id="1" Degree="0.0"/>
36        </Rules>
37      </Resource>
38      <Resource Id="14" Value="0.46" Level="0.64">
39        <Rules>
40          <Rule Id="3" Degree="0.23"/>
41          <Rule Id="2" Degree="0.0"/>
42          <Rule Id="1" Degree="0.0"/>
43        </Rules>
44      </Resource>
45      <Resource Id="13" Value="0.43" Level="0.64">
46        <Rules>
47          <Rule Id="3" Degree="0.19"/>
48          <Rule Id="2" Degree="0.0"/>
49          <Rule Id="1" Degree="0.0"/>
50        </Rules>
51      </Resource>
52      <Resource Id="12" Value="0.47" Level="0.65">
53        <Rules>
54          <Rule Id="3" Degree="0.24"/>
55          <Rule Id="2" Degree="0.0"/>
56          <Rule Id="1" Degree="0.0"/>
57        </Rules>
58      </Resource>
59      <Resource Id="11" Value="0.46" Level="0.64">
60        <Rules>
61          <Rule Id="3" Degree="0.23"/>
62          <Rule Id="2" Degree="0.0"/>
63          <Rule Id="1" Degree="0.0"/>
64        </Rules>
65      </Resource>
66      <Resource Id="10" Value="0.46" Level="0.64">
67        <Rules>
68          <Rule Id="3" Degree="0.23"/>
69          <Rule Id="2" Degree="0.0"/>
70          <Rule Id="1" Degree="0.0"/>
```

```
71        </Rules>
72      </Resource>
73      <Resource Id="9" Value="0.38" Level="0.62">
74        <Rules>
75          <Rule Id="3" Degree="0.11"/>
76          <Rule Id="2" Degree="0.0"/>
77          <Rule Id="1" Degree="0.0"/>
78        </Rules>
79      </Resource>
80      <Resource Id="8" Value="0.22" Level="0.3">
81        <Rules>
82          <Rule Id="3" Degree="0.0"/>
83          <Rule Id="2" Degree="0.8"/>
84          <Rule Id="1" Degree="0.0"/>
85        </Rules>
86      </Resource>
87      <Resource Id="7" Value="0.46" Level="0.64">
88        <Rules>
89          <Rule Id="3" Degree="0.23"/>
90          <Rule Id="2" Degree="0.0"/>
91          <Rule Id="1" Degree="0.0"/>
92        </Rules>
93      </Resource>
94      <Resource Id="6" Value="0.37" Level="0.62">
95        <Rules>
96          <Rule Id="3" Degree="0.1"/>
97          <Rule Id="2" Degree="0.0"/>
98          <Rule Id="1" Degree="0.0"/>
99        </Rules>
100     </Resource>
101     <Resource Id="5" Value="0.46" Level="0.64">
102       <Rules>
103         <Rule Id="3" Degree="0.23"/>
104         <Rule Id="2" Degree="0.0"/>
105         <Rule Id="1" Degree="0.0"/>
106       </Rules>
107     </Resource>
108     <Resource Id="4" Value="0.46" Level="0.64">
109       <Rules>
110         <Rule Id="3" Degree="0.23"/>
111         <Rule Id="2" Degree="0.0"/>
112         <Rule Id="1" Degree="0.0"/>
113       </Rules>
114     </Resource>
```

```
115     <Resource Id="3" Value="0.46" Level="0.64">
116       <Rules>
117         <Rule Id="3" Degree="0.23"/>
118         <Rule Id="2" Degree="0.0"/>
119         <Rule Id="1" Degree="0.0"/>
120       </Rules>
121     </Resource>
122     <Resource Id="2" Value="0.48" Level="0.65">
123       <Rules>
124         <Rule Id="3" Degree="0.26"/>
125         <Rule Id="2" Degree="0.0"/>
126         <Rule Id="1" Degree="0.0"/>
127       </Rules>
128     </Resource>
129     <Resource Id="22" Value="0.46" Level="0.64">
130       <Rules>
131         <Rule Id="3" Degree="0.23"/>
132         <Rule Id="2" Degree="0.0"/>
133         <Rule Id="1" Degree="0.0"/>
134       </Rules>
135     </Resource>
136     <Resource Id="1" Value="0.42" Level="0.63">
137       <Rules>
138         <Rule Id="3" Degree="0.17"/>
139         <Rule Id="2" Degree="0.0"/>
140         <Rule Id="1" Degree="0.0"/>
141       </Rules>
142     </Resource>
143     <Resource Id="21" Value="0.46" Level="0.64">
144       <Rules>
145         <Rule Id="3" Degree="0.23"/>
146         <Rule Id="2" Degree="0.0"/>
147         <Rule Id="1" Degree="0.0"/>
148       </Rules>
149     </Resource>
150     <Resource Id="20" Value="0.45" Level="0.64">
151       <Rules>
152         <Rule Id="3" Degree="0.21"/>
153         <Rule Id="2" Degree="0.0"/>
154         <Rule Id="1" Degree="0.0"/>
155       </Rules>
156     </Resource>
157   </Resources>
158 </Report>
```

Sample MDS File

```
<?xml version="1.0" encoding="UTF-8"?>
<context-group id="100" name="GridPP">
  <context id="1" name="EFDA-JET" />
  <context id="2" name="RAL-LCG2_Tier-1" />
  <context-group id="101" name="LT2">
    <context id="3" name="Brunel"/>
    <context id="4" name="IC-HEP" />
    <context id="5" name="IC-LeSC" />
    <context id="6" name="QMUL" />
    <context id="7" name="RHUL" />
    <context id="8" name="UCL-CENTRAL" />
    <context id="9" name="UCL-HEP" />
  </context-group>
  <context-group id="102" name="NORTHGRID">
    <context id="10" name="LANCS-HEP" />
    <context id="11" name="LIV-HEP" />
    <context id="12" name="MAN-HEP" />
    <context id="13" name="SHEF-HE" />
  </context-group>
  <context-group id="103" name="SCOTGRID">
    <context id="14" name="DURHAM" />
    <context id="15" name="ECDF" />
    <context id="16" name="GLASGOW" />
  </context-group>
  <context-group id="104" name="SOUTHGRID">
    <context id="17" name="BHAM-HEP" />
    <context id="18" name="BRIS-HEP" />
    <context id="19" name="CAM-HEP" />
    <context id="20" name="OX-HEP" />
    <context id="21" name="RALPP" />
  </context-group>
  <context id="22" name="csTCDie␣" />
</context-group>
```

.2.3 Sample Outputs

GREPTrust usage:

```
java -jar greptrust.jar [OPTIONS]
```

Option	Switch	Input Mask	Optional
CLIENT_ID	`-c, --client`	\d{1,}	no
RESOURCE_ID(s)	`-r, --resource`	\d{1,} \,?	no
TDS_FILE	`-f, --file`	\[\w \\\.] {1,}	no
CTX_VALUE	`-v, --context`	\[01] \. \d	no
OUTPUT_FILE	`-o, --output`	\[\w \\\.] {1,}	yes
CUTOFF_DATE	`-d, --date`	\d{8}	yes
TDF	`-t, --tdf`	\d{1,}	yes
SHOW_HELP	`-h, --help`	-	yes

Table 1: Command line options

Performing a Reputation-Policy Query on set of Grid resources

```
C:\> java -jar greptrust.jar -c 1 -r 1,2,3,4,5,6,7,8,9,10,
11,12,13,14,15,16,17,18,19,20,21,22 -f tds1.xml -o output.xml

2009-10-26 20:02:45,256 [main] INFO: process:
:RA: Received Reputation-Policy Query
2009-10-26 20:02:45,283 [main] INFO: process:
:RA: RPQ is: C1R[Ljava.lang.String;@1ad086a
2009-10-26 20:02:45,847 [main] INFO: process:
:RA: clientId=1
2009-10-26 20:02:45,847 [main] INFO: process:
:RA: resources=1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
2009-10-26 20:02:45,848 [main] INFO: process:
:RA: cutoffDateTime=NULL
2009-10-26 20:02:45,848 [main] INFO: process:
:RA: trustDecayFunction=0
2009-10-26 20:02:45,870 [main] INFO: <init>:
:OA: Initialized Opinion Accumulator.
2009-10-26 20:02:45,884 [main] INFO: <init>:
:CMP: Initialized Customized Matrices Pool
2009-10-26 20:02:45,885 [main] INFO: <init>:
:CP: (1.8525058) Initialized Correlation Process.
2009-10-26 20:02:45,886 [main] INFO: process:
:CP: (1.8525058) Processing opinions.
2009-10-26 20:02:45,891 [main] INFO: <init>:
:CP: (1.8525058.16779281) Initialized Child Correlation Process.
```

```
2009-10-26 20:02:45,893 [Thread-1] INFO: run:
:CCP: (1.8525058.16779281) Forked Child CP. [O=1]
2009-10-26 20:02:45,893 [Thread-1] INFO: getOpinionMatrix:
:CMP: Contacted Customized Matrices Pool to return OM for opinion: 1
2009-10-26 20:02:45,903 [Thread-1] INFO: solve:
:OM: Generating OM=CALL OPINION_MATRIX_SELECT(1, NULL, 0);
2009-10-26 20:02:46,002 [Thread-1] DEBUG: locate:
ConfigurationUtils.locate(): base is null, name is
C:\Users\Yonatan\workspace\GREPTrustAnalysis\cfg\SystemConfig.xml
2009-10-26 20:02:46,004 [Thread-1] DEBUG: locate:
Loading configuration from the absolute path
C:\Users\Yonatan\workspace\GREPTrustAnalysis\cfg\SystemConfig.xml
2009-10-26 20:06:06,962 [Thread-1] INFO: solve: :OM: C=1,R=1,V=0.42
2009-10-26 20:06:06,986 [Thread-1] INFO: solve: :OM: C=1,R=2,V=0.48
2009-10-26 20:06:06,987 [Thread-1] INFO: solve: :OM: C=1,R=3,V=0.47
2009-10-26 20:06:06,987 [Thread-1] INFO: solve: :OM: C=1,R=4,V=0.47
2009-10-26 20:06:06,987 [Thread-1] INFO: solve: :OM: C=1,R=5,V=0.46
2009-10-26 20:06:06,987 [Thread-1] INFO: solve: :OM: C=1,R=6,V=0.37
2009-10-26 20:06:06,988 [Thread-1] INFO: solve: :OM: C=1,R=7,V=0.46
2009-10-26 20:06:06,988 [Thread-1] INFO: solve: :OM: C=1,R=8,V=0.22
2009-10-26 20:06:06,988 [Thread-1] INFO: solve: :OM: C=1,R=9,V=0.38
2009-10-26 20:06:06,988 [Thread-1] INFO: solve: :OM: C=1,R=10,V=0.46
2009-10-26 20:06:06,989 [Thread-1] INFO: solve: :OM: C=1,R=11,V=0.46
2009-10-26 20:06:06,989 [Thread-1] INFO: solve: :OM: C=1,R=12,V=0.47
2009-10-26 20:06:06,989 [Thread-1] INFO: solve: :OM: C=1,R=13,V=0.43
2009-10-26 20:06:06,990 [Thread-1] INFO: solve: :OM: C=1,R=14,V=0.46
2009-10-26 20:06:06,990 [Thread-1] INFO: solve: :OM: C=1,R=15,V=0.38
2009-10-26 20:06:06,990 [Thread-1] INFO: solve: :OM: C=1,R=16,V=0.48
2009-10-26 20:06:06,990 [Thread-1] INFO: solve: :OM: C=1,R=17,V=0.48
2009-10-26 20:06:06,990 [Thread-1] INFO: solve: :OM: C=1,R=18,V=0.49
2009-10-26 20:06:06,991 [Thread-1] INFO: solve: :OM: C=1,R=19,V=0.42
2009-10-26 20:06:06,992 [Thread-1] INFO: solve: :OM: C=1,R=20,V=0.45
2009-10-26 20:06:06,993 [Thread-1] INFO: solve: :OM: C=1,R=21,V=0.47
2009-10-26 20:06:06,993 [Thread-1] INFO: solve: :OM: C=1,R=22,V=0.46
2009-10-26 20:06:06,993 [Thread-1] INFO: solve: :OM: C=2,R=1,V=0.41
2009-10-26 20:06:06,994 [Thread-1] INFO: solve: :OM: C=2,R=2,V=0.48
2009-10-26 20:06:06,994 [Thread-1] INFO: solve: :OM: C=2,R=3,V=0.46
2009-10-26 20:06:06,994 [Thread-1] INFO: solve: :OM: C=2,R=4,V=0.46
2009-10-26 20:06:06,994 [Thread-1] INFO: solve: :OM: C=2,R=5,V=0.45
2009-10-26 20:06:06,994 [Thread-1] INFO: solve: :OM: C=2,R=6,V=0.36
2009-10-26 20:06:06,995 [Thread-1] INFO: solve: :OM: C=2,R=7,V=0.45
2009-10-26 20:06:06,995 [Thread-1] INFO: solve: :OM: C=2,R=8,V=0.22
2009-10-26 20:06:06,995 [Thread-1] INFO: solve: :OM: C=2,R=9,V=0.37
2009-10-26 20:06:06,995 [Thread-1] INFO: solve: :OM: C=2,R=10,V=0.45
```

```
2009-10-26 20:06:06,995 [Thread-1] INFO: solve: :OM: C=2,R=11,V=0.46
2009-10-26 20:06:06,996 [Thread-1] INFO: solve: :OM: C=2,R=12,V=0.47
2009-10-26 20:06:06,996 [Thread-1] INFO: solve: :OM: C=2,R=13,V=0.42
2009-10-26 20:06:06,996 [Thread-1] INFO: solve: :OM: C=2,R=14,V=0.46
2009-10-26 20:06:06,996 [Thread-1] INFO: solve: :OM: C=2,R=15,V=0.37
2009-10-26 20:06:06,999 [Thread-1] INFO: solve: :OM: C=2,R=16,V=0.47
2009-10-26 20:06:06,999 [Thread-1] INFO: solve: :OM: C=2,R=17,V=0.47
2009-10-26 20:06:06,999 [Thread-1] INFO: solve: :OM: C=2,R=18,V=0.49
2009-10-26 20:06:06,999 [Thread-1] INFO: solve: :OM: C=2,R=19,V=0.41
2009-10-26 20:06:07,000 [Thread-1] INFO: solve: :OM: C=2,R=20,V=0.44
2009-10-26 20:06:07,000 [Thread-1] INFO: solve: :OM: C=2,R=21,V=0.46
2009-10-26 20:06:07,000 [Thread-1] INFO: solve: :OM: C=2,R=22,V=0.46
2009-10-26 20:06:07,041 [Thread-1] INFO: run:
:CCP: (1.8525058.16779281) Received Opinion tuple from OM.
[O=1;R=19,V=0.42;R=18,V=0.49;R=17,V=0.48;R=16,V=0.48;R=15,V=0.38;
R=14,V=0.46;R=13,V=0.43;R=12,V=0.47;R=11,V=0.46;R=10,V=0.46;
R=9,V=0.38;R=8,V=0.22;R=7,V=0.46;
R=6,V=0.37;R=5,V=0.46;R=4,V=0.46;R=3,V=0.46;R=2,V=0.48;
R=1,V=0.42;R=22,V=0.46;R=21,V=0.46;R=20,V=0.45]
2009-10-26 20:06:07,041 [Thread-1] INFO: put: :OA:
Inserting Opinion Tuple into OA. (O=1,W=1.0)
2009-10-26 20:06:07,043 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=19,VxW=0.42]
2009-10-26 20:06:07,043 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=18,VxW=0.49]
2009-10-26 20:06:07,044 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=17,VxW=0.48]
2009-10-26 20:06:07,044 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=16,VxW=0.48]
2009-10-26 20:06:07,044 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=15,VxW=0.38]
2009-10-26 20:06:07,045 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=14,VxW=0.46]
2009-10-26 20:06:07,045 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=13,VxW=0.43]
2009-10-26 20:06:07,046 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=12,VxW=0.47]
2009-10-26 20:06:07,046 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=11,VxW=0.46]
2009-10-26 20:06:07,046 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=10,VxW=0.46]
2009-10-26 20:06:07,047 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=9,VxW=0.38]
2009-10-26 20:06:07,047 [Thread-1] INFO: put: :OA:
```

```
Inserting values into OA. [R=8,VxW=0.22]
2009-10-26 20:06:07,048 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=7,VxW=0.46]
2009-10-26 20:06:07,048 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=6,VxW=0.37]
2009-10-26 20:06:07,048 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=5,VxW=0.46]
2009-10-26 20:06:07,049 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=4,VxW=0.46]
2009-10-26 20:06:07,049 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=3,VxW=0.46]
2009-10-26 20:06:07,049 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=2,VxW=0.48]
2009-10-26 20:06:07,050 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=1,VxW=0.42]
2009-10-26 20:06:07,050 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=22,VxW=0.46]
2009-10-26 20:06:07,050 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=21,VxW=0.46]
2009-10-26 20:06:07,052 [Thread-1] INFO: put: :OA:
Inserting values into OA. [R=20,VxW=0.45]
2009-10-26 20:06:07,052 [main] INFO: process: :CP:
(1.8525058) All Child CP forks returned. Performing summary on OA.
2009-10-26 20:06:07,678 [main] INFO: process: :RA:
Initialized TDR Engine. contextValue=-1.0
2009-10-26 20:06:07,734 [main] INFO: process:
:RA: Generated Resource Value Table
2009-10-26 20:06:07,735 [main] INFO: process: :RA: R=19,V=0.42
2009-10-26 20:06:07,798 [main] INFO: process: :DRE: Rule=3,Degree=0.17
2009-10-26 20:06:07,819 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,819 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,820 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.63
2009-10-26 20:06:07,820 [main] INFO: process: :RA: R=18,V=0.49
2009-10-26 20:06:07,821 [main] INFO: process: :DRE: Rule=3,Degree=0.27
2009-10-26 20:06:07,821 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,822 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,822 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.65
2009-10-26 20:06:07,824 [main] INFO: process: :RA: R=17,V=0.48
2009-10-26 20:06:07,825 [main] INFO: process: :DRE: Rule=3,Degree=0.26
2009-10-26 20:06:07,828 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,828 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,828 [main] INFO: process:
```

```
:DRE: Rule=trust_level,Degree=0.65
2009-10-26 20:06:07,829 [main] INFO: process: :RA: R=16,V=0.48
2009-10-26 20:06:07,829 [main] INFO: process: :DRE: Rule=3,Degree=0.26
2009-10-26 20:06:07,830 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,832 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,832 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.65
2009-10-26 20:06:07,832 [main] INFO: process: :RA: R=15,V=0.38
2009-10-26 20:06:07,833 [main] INFO: process: :DRE: Rule=3,Degree=0.11
2009-10-26 20:06:07,833 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,834 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,834 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.62
2009-10-26 20:06:07,834 [main] INFO: process: :RA: R=14,V=0.46
2009-10-26 20:06:07,835 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,835 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,836 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,836 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,836 [main] INFO: process: :RA: R=13,V=0.43
2009-10-26 20:06:07,837 [main] INFO: process: :DRE: Rule=3,Degree=0.19
2009-10-26 20:06:07,837 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,837 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,838 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,838 [main] INFO: process: :RA: R=12,V=0.47
2009-10-26 20:06:07,839 [main] INFO: process: :DRE: Rule=3,Degree=0.24
2009-10-26 20:06:07,839 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,839 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,841 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.65
2009-10-26 20:06:07,841 [main] INFO: process: :RA: R=11,V=0.46
2009-10-26 20:06:07,843 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,844 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,844 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,845 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,845 [main] INFO: process: :RA: R=10,V=0.46
2009-10-26 20:06:07,846 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,846 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,846 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,848 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,848 [main] INFO: process: :RA: R=9,V=0.38
```

```
2009-10-26 20:06:07,849 [main] INFO: process: :DRE: Rule=3,Degree=0.11
2009-10-26 20:06:07,849 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,849 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,850 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.62
2009-10-26 20:06:07,850 [main] INFO: process: :RA: R=8,V=0.22
2009-10-26 20:06:07,850 [main] INFO: process: :DRE: Rule=3,Degree=0.0
2009-10-26 20:06:07,850 [main] INFO: process: :DRE: Rule=2,Degree=0.8
2009-10-26 20:06:07,851 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,851 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.3
2009-10-26 20:06:07,851 [main] INFO: process: :RA: R=7,V=0.46
2009-10-26 20:06:07,851 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,852 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,852 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,852 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,852 [main] INFO: process: :RA: R=6,V=0.37
2009-10-26 20:06:07,853 [main] INFO: process: :DRE: Rule=3,Degree=0.1
2009-10-26 20:06:07,853 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,853 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,854 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.62
2009-10-26 20:06:07,854 [main] INFO: process: :RA: R=5,V=0.46
2009-10-26 20:06:07,855 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,855 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,855 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,856 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,856 [main] INFO: process: :RA: R=4,V=0.46
2009-10-26 20:06:07,856 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,857 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,857 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,857 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,858 [main] INFO: process: :RA: R=3,V=0.46
2009-10-26 20:06:07,858 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,859 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,859 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,859 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,859 [main] INFO: process: :RA: R=2,V=0.48
2009-10-26 20:06:07,860 [main] INFO: process: :DRE: Rule=3,Degree=0.26
2009-10-26 20:06:07,860 [main] INFO: process: :DRE: Rule=2,Degree=0.0
```

```
2009-10-26 20:06:07,860 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,860 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.65
2009-10-26 20:06:07,860 [main] INFO: process: :RA: R=22,V=0.46
2009-10-26 20:06:07,861 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,861 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,861 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,862 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,862 [main] INFO: process: :RA: R=1,V=0.42
2009-10-26 20:06:07,862 [main] INFO: process: :DRE: Rule=3,Degree=0.17
2009-10-26 20:06:07,863 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,863 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,863 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.63
2009-10-26 20:06:07,863 [main] INFO: process: :RA: R=21,V=0.46
2009-10-26 20:06:07,864 [main] INFO: process: :DRE: Rule=3,Degree=0.23
2009-10-26 20:06:07,864 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,864 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,865 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,865 [main] INFO: process: :RA: R=20,V=0.45
2009-10-26 20:06:07,866 [main] INFO: process: :DRE: Rule=3,Degree=0.21
2009-10-26 20:06:07,866 [main] INFO: process: :DRE: Rule=2,Degree=0.0
2009-10-26 20:06:07,866 [main] INFO: process: :DRE: Rule=1,Degree=0.0
2009-10-26 20:06:07,867 [main] INFO: process:
:DRE: Rule=trust_level,Degree=0.64
2009-10-26 20:06:07,995 [main] INFO: process: :RA: Report generated.
```

Performing a Reputation-Policy Query on set on a VO

```
C:\> java -jar greptrust.jar -c 1 -r 100 -f tds1.xml -o output.xml

processItems() - Invoking greptrust: 2 104 [17, 18, 19, 20, 21]
2009-09-19 20:58:59,843 [main] DEBUG: getInstance: getInstance() - start
2009-09-19 20:58:59,843 [main] DEBUG: getInstance: getInstance() - end
2009-09-19 20:58:59,843 [main] INFO: allocate:
allocate() - IQueryManager allocating resource
uri:n47.cpc.wmin.ac.uk:2154/GREPTrust
2009-09-19 20:58:59,843 [main] DEBUG: query:
query(IReputationQueryContext) - start
2009-09-19 20:58:59,968 [main] INFO: getTrustValue:
getTrustValue(String) - resource=17 trustValue=0.65 deterministicMode=true
2009-09-19 20:58:59,984 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=17 trustValue=0.65
```

```
2009-09-19 20:58:59,984 [main] INFO: getTrustValue:
getTrustValue(String) - resource=18 trustValue=0.42 deterministicMode=true
2009-09-19 20:58:59,984 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=18 trustValue=0.42
2009-09-19 20:58:59,984 [main] INFO: getTrustValue:
getTrustValue(String) - resource=19 trustValue=0.13 deterministicMode=true
2009-09-19 20:58:59,984 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=19 trustValue=0.13
2009-09-19 20:58:59,984 [main] INFO: getTrustValue:
getTrustValue(String) - resource=20 trustValue=0.77 deterministicMode=true
2009-09-19 20:59:00,000 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=20 trustValue=0.77
2009-09-19 20:59:00,000 [main] INFO: getTrustValue:
getTrustValue(String) - resource=21 trustValue=0.23 deterministicMode=true
2009-09-19 20:59:00,000 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=21 trustValue=0.23
2009-09-19 20:59:00,234 [main] DEBUG: query:
query(IReputationQueryContext) - end
2009-09-19 20:59:00,343 [main] INFO: processEvaluationModel:
processItems() - double mean=0.44
2009-09-19 20:59:00,343 [main] DEBUG: processEvaluationModel:
put() key is new - 2 adding mean:0.44
2009-09-19 20:59:00,343 [main] DEBUG: processEvaluationModel:

processItems() - Invoking greptrust: 2 103 [14, 15, 16]
2009-09-19 20:59:00,343 [main] DEBUG: getInstance: getInstance() - start
2009-09-19 20:59:00,343 [main] DEBUG: getInstance: getInstance() - end
2009-09-19 20:59:00,343 [main] INFO: allocate:
allocate() - IQueryManager allocating resource
uri:n42.cpc.wmin.ac.uk:2165/GREPTrust
2009-09-19 20:59:00,359 [main] DEBUG: query:
query(IReputationQueryContext) - start
2009-09-19 20:59:00,515 [main] INFO: getTrustValue:
getTrustValue(String) - resource=14 trustValue=0.76 deterministicMode=true
2009-09-19 20:59:00,515 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=14 trustValue=0.76
2009-09-19 20:59:00,515 [main] INFO: getTrustValue:
getTrustValue(String) - resource=15 trustValue=0.78 deterministicMode=true
2009-09-19 20:59:00,515 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=15 trustValue=0.78
2009-09-19 20:59:00,531 [main] INFO: getTrustValue:
getTrustValue(String) - resource=16 trustValue=0.23 deterministicMode=true
2009-09-19 20:59:00,531 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=16 trustValue=0.23
```

```
2009-09-19 20:59:00,671 [main] DEBUG: query:
query(IReputationQueryContext) - end
2009-09-19 20:59:00,718 [main] INFO: processEvaluationModel:
processItems() - double mean=0.59
2009-09-19 20:59:00,734 [main] DEBUG: processEvaluationModel:
put() key exists - 2 adding mean:0.59
2009-09-19 20:59:00,734 [main] DEBUG: processEvaluationModel:

processItems() - Invoking greptrust: 2 102 [10, 11, 12, 13]
2009-09-19 20:59:00,734 [main] DEBUG: getInstance: getInstance() - start
2009-09-19 20:59:00,734 [main] DEBUG: getInstance: getInstance() - end
2009-09-19 20:59:00,734 [main] INFO: allocate:
allocate() - IQueryManager allocating resource
uri:n44.cpc.wmin.ac.uk:2111/GREPTrust
2009-09-19 20:59:00,734 [main] DEBUG: query:
query(IReputationQueryContext) - start
2009-09-19 20:59:00,828 [main] INFO: getTrustValue:
getTrustValue(String) - resource=10 trustValue=0.55 deterministicMode=true
2009-09-19 20:59:00,843 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=10 trustValue=0.55
2009-09-19 20:59:00,843 [main] INFO: getTrustValue:
getTrustValue(String) - resource=11 trustValue=0.67 deterministicMode=true
2009-09-19 20:59:00,843 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=11 trustValue=0.67
2009-09-19 20:59:00,843 [main] INFO: getTrustValue:
getTrustValue(String) - resource=12 trustValue=0.87 deterministicMode=true
2009-09-19 20:59:00,843 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=12 trustValue=0.87
2009-09-19 20:59:00,843 [main] INFO: getTrustValue:
getTrustValue(String) - resource=13 trustValue=0.34 deterministicMode=true
2009-09-19 20:59:00,843 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=13 trustValue=0.34
2009-09-19 20:59:01,015 [main] DEBUG: query:
query(IReputationQueryContext) - end
2009-09-19 20:59:01,046 [main] INFO: processEvaluationModel:
processItems() - double mean=0.61
2009-09-19 20:59:01,062 [main] DEBUG: processEvaluationModel:
put() key exists - 2 adding mean:0.61
2009-09-19 20:59:01,062 [main] DEBUG: processEvaluationModel:

processItems() - Invoking greptrust: 2 101 [3, 4, 5, 6, 7, 8, 9]
2009-09-19 20:59:01,062 [main] DEBUG: getInstance: getInstance() - start
2009-09-19 20:59:01,062 [main] DEBUG: getInstance: getInstance() - end
2009-09-19 20:59:01,078 [main] INFO: allocate:
```

```
allocate() - IQueryManager allocating resource
uri:n49.cpc.wmin.ac.uk:2190/GREPTrust
2009-09-19 20:59:01,078 [main] DEBUG: query:
query(IReputationQueryContext) - start
2009-09-19 20:59:01,218 [main] INFO: getTrustValue:
getTrustValue(String) - resource=3 trustValue=0.35 deterministicMode=true
2009-09-19 20:59:01,234 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=3 trustValue=0.35
2009-09-19 20:59:01,234 [main] INFO: getTrustValue:
getTrustValue(String) - resource=4 trustValue=0.45 deterministicMode=true
2009-09-19 20:59:01,234 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=4 trustValue=0.45
2009-09-19 20:59:01,234 [main] INFO: getTrustValue:
getTrustValue(String) - resource=5 trustValue=0.46 deterministicMode=true
2009-09-19 20:59:01,234 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=5 trustValue=0.46
2009-09-19 20:59:01,234 [main] INFO: getTrustValue:
getTrustValue(String) - resource=6 trustValue=0.47 deterministicMode=true
2009-09-19 20:59:01,234 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=6 trustValue=0.47
2009-09-19 20:59:01,234 [main] INFO: getTrustValue:
getTrustValue(String) - resource=7 trustValue=0.48 deterministicMode=true
2009-09-19 20:59:01,250 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=7 trustValue=0.48
2009-09-19 20:59:01,250 [main] INFO: getTrustValue:
getTrustValue(String) - resource=8 trustValue=0.49 deterministicMode=true
2009-09-19 20:59:01,250 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=8 trustValue=0.49
2009-09-19 20:59:01,250 [main] INFO: getTrustValue:
getTrustValue(String) - resource=9 trustValue=0.5 deterministicMode=true
2009-09-19 20:59:01,265 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=9 trustValue=0.5
2009-09-19 20:59:01,468 [main] DEBUG: query:
query(IReputationQueryContext) - end
2009-09-19 20:59:01,546 [main] INFO: processEvaluationModel:
processItems() - double mean=0.46
2009-09-19 20:59:01,546 [main] DEBUG: processEvaluationModel:
put() key exists - 2 adding mean:0.46
2009-09-19 20:59:01,546 [main] DEBUG: processEvaluationModel:

processItems() - Invoking greptrust: 1 100 [1, 2, 22]
2009-09-19 20:59:01,546 [main] DEBUG: getInstance: getInstance() - start
2009-09-19 20:59:01,546 [main] DEBUG: getInstance: getInstance() - end
2009-09-19 20:59:01,562 [main] INFO: allocate:
```

```
allocate() - IQueryManager allocating resource
uri:n46.cpc.wmin.ac.uk:2189/GREPTrust
2009-09-19 20:59:01,562 [main] DEBUG: query:
query(IReputationQueryContext) - start
2009-09-19 20:59:01,687 [main] INFO: getTrustValue:
getTrustValue(String) - resource=1 trustValue=0.15 deterministicMode=true
2009-09-19 20:59:01,703 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=1 trustValue=0.15
2009-09-19 20:59:01,703 [main] INFO: getTrustValue:
getTrustValue(String) - resource=2 trustValue=0.25 deterministicMode=true
2009-09-19 20:59:01,703 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=2 trustValue=0.25
2009-09-19 20:59:01,703 [main] INFO: getTrustValue:
getTrustValue(String) - resource=22 trustValue=0.97 deterministicMode=true
2009-09-19 20:59:01,703 [main] INFO: query:
query(IReputationQueryContext) - Assigning resource=22 trustValue=0.97
2009-09-19 20:59:01,796 [main] DEBUG: query:
query(IReputationQueryContext) - end
2009-09-19 20:59:01,875 [main] INFO: processEvaluationModel:
processItems() - double mean=0.46
2009-09-19 20:59:01,875 [main] DEBUG: processEvaluationModel:
put() key is new - 1 adding mean:0.46
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:

processItems() - process depth mean values
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - Integer[] dmvKeys=2
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - Integer meanDepth=2
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - ArrayList<Double> values=[0.44, 0.59, 0.61, 0.46]
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - double totalMean=0.53
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - Integer meanDepth=1
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - ArrayList<Double> values=[0.46]
2009-09-19 20:59:01,890 [main] INFO: processEvaluationModel:
processItems() - double totalMean=0.46
2009-09-19 20:59:01,906 [main] INFO: processEvaluationModel:
processItems() - context trustValue=0.5
2009-09-19 20:59:02,078 [main] DEBUG: query:
query(IReputationQueryContext) - end
```

Resource Inference

```
C:\> java -jar greptrust.jar -v 0.45 -c 1 -r 1,2,3,4,5,6,7,8,9,10,
11,12,13,14,15,16,17,18,19,20,21,22 -f tds-2opinions.xml -o output.xml

2009-11-15 12:29:33,869 [Thread-9] INFO: process:
:RA: Received Reputation-Policy Query
2009-11-15 12:29:33,870 [Thread-9] INFO: process:
:RA: RPQ is: C1R[Ljava.lang.String;@141b571
2009-11-15 12:29:33,894 [Thread-9] INFO: process: :RA: clientId=1
2009-11-15 12:29:33,894 [Thread-9] INFO: process:
:RA: resources=1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
2009-11-15 12:29:33,894 [Thread-9] INFO: process: :RA: cutoffDateTime=NULL
2009-11-15 12:29:33,894 [Thread-9] INFO: process: :RA: trustDecayFunction=0
2009-11-15 12:29:33,895 [Thread-9] INFO: <init>:
:OA: Initialized Opinion Accumulator.
2009-11-15 12:29:33,895 [Thread-9] INFO: <init>:
:CP: (1.4199185) Initialized Correlation Process.
2009-11-15 12:29:33,896 [Thread-9] INFO: process:
:CP: (1.4199185) Processing opinions.
2009-11-15 12:29:33,897 [Thread-9] INFO: <init>:
:CP: (1.4199185.26214498) Initialized Child Correlation Process.
2009-11-15 12:29:33,897 [Thread-19] INFO: run:
:CCP: (1.4199185.26214498) Forked Child CP. [O=1]
2009-11-15 12:29:33,898 [Thread-19] INFO: getOpinionMatrix:
:CMP: Contacted Customized Metrics Pool to return OM for opinion: 1
2009-11-15 12:29:33,898 [Thread-19] INFO: solve:
:OM: Generating OM=CALL OPINION_MATRIX_SELECT(1, NULL, 0);
2009-11-15 12:41:44,253 [Thread-19] INFO: solve: :OM: C=1,R=1,V=0.12
2009-11-15 12:41:44,253 [Thread-19] INFO: solve: :OM: C=1,R=2,V=0.41
2009-11-15 12:41:44,253 [Thread-19] INFO: solve: :OM: C=1,R=3,V=0.52
2009-11-15 12:41:44,253 [Thread-19] INFO: solve: :OM: C=1,R=4,V=0.17
2009-11-15 12:41:44,254 [Thread-19] INFO: solve: :OM: C=1,R=5,V=0.23
2009-11-15 12:41:44,254 [Thread-19] INFO: solve: :OM: C=1,R=6,V=0.23
2009-11-15 12:41:44,254 [Thread-19] INFO: solve: :OM: C=1,R=7,V=0.66
2009-11-15 12:41:44,254 [Thread-19] INFO: solve: :OM: C=1,R=8,V=0.39
2009-11-15 12:41:44,254 [Thread-19] INFO: solve: :OM: C=1,R=9,V=0.53
2009-11-15 12:41:44,258 [Thread-19] INFO: solve: :OM: C=1,R=10,V=0.70
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=11,V=0.43
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=12,V=0.66
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=13,V=0.15
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=14,V=0.57
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=15,V=0.42
2009-11-15 12:41:44,259 [Thread-19] INFO: solve: :OM: C=1,R=16,V=0.54
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=17,V=0.22
```

```
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=18,V=0.50
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=19,V=0.66
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=20,V=0.24
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=21,V=0.01
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=1,R=22,V=0.22
2009-11-15 12:41:44,260 [Thread-19] INFO: solve: :OM: C=2,R=1,V=0.12
2009-11-15 12:41:44,261 [Thread-19] INFO: solve: :OM: C=2,R=2,V=0.41
2009-11-15 12:41:44,261 [Thread-19] INFO: solve: :OM: C=2,R=3,V=0.52
2009-11-15 12:41:44,262 [Thread-19] INFO: solve: :OM: C=2,R=4,V=0.17
2009-11-15 12:41:44,263 [Thread-19] INFO: solve: :OM: C=2,R=5,V=0.23
2009-11-15 12:41:44,263 [Thread-19] INFO: solve: :OM: C=2,R=6,V=0.23
2009-11-15 12:41:44,263 [Thread-19] INFO: solve: :OM: C=2,R=7,V=0.66
2009-11-15 12:41:44,263 [Thread-19] INFO: solve: :OM: C=2,R=8,V=0.39
2009-11-15 12:41:44,265 [Thread-19] INFO: solve: :OM: C=2,R=9,V=0.53
2009-11-15 12:41:44,265 [Thread-19] INFO: solve: :OM: C=2,R=10,V=0.70
2009-11-15 12:41:44,265 [Thread-19] INFO: solve: :OM: C=2,R=11,V=0.43
2009-11-15 12:41:44,265 [Thread-19] INFO: solve: :OM: C=2,R=12,V=0.66
2009-11-15 12:41:44,265 [Thread-19] INFO: solve: :OM: C=2,R=13,V=0.15
2009-11-15 12:41:44,266 [Thread-19] INFO: solve: :OM: C=2,R=14,V=0.57
2009-11-15 12:41:44,266 [Thread-19] INFO: solve: :OM: C=2,R=15,V=0.42
2009-11-15 12:41:44,266 [Thread-19] INFO: solve: :OM: C=2,R=16,V=0.54
2009-11-15 12:41:44,266 [Thread-19] INFO: solve: :OM: C=2,R=17,V=0.22
2009-11-15 12:41:44,272 [Thread-19] INFO: solve: :OM: C=2,R=18,V=0.50
2009-11-15 12:41:44,272 [Thread-19] INFO: solve: :OM: C=2,R=19,V=0.66
2009-11-15 12:41:44,274 [Thread-19] INFO: solve: :OM: C=2,R=20,V=0.24
2009-11-15 12:41:44,274 [Thread-19] INFO: solve: :OM: C=2,R=21,V=0.01
2009-11-15 12:41:44,274 [Thread-19] INFO: solve: :OM: C=2,R=22,V=0.22
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=1,V=0.12
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=2,V=0.41
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=3,V=0.52
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=4,V=0.17
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=5,V=0.23
2009-11-15 12:41:44,275 [Thread-19] INFO: solve: :OM: C=3,R=6,V=0.23
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=7,V=0.66
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=8,V=0.39
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=9,V=0.53
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=10,V=0.70
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=11,V=0.43
2009-11-15 12:41:44,276 [Thread-19] INFO: solve: :OM: C=3,R=12,V=0.66
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=13,V=0.15
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=14,V=0.57
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=15,V=0.42
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=16,V=0.54
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=17,V=0.22
```

```
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=18,V=0.50
2009-11-15 12:41:44,277 [Thread-19] INFO: solve: :OM: C=3,R=19,V=0.66
2009-11-15 12:41:44,278 [Thread-19] INFO: solve: :OM: C=3,R=20,V=0.24
2009-11-15 12:41:44,278 [Thread-19] INFO: solve: :OM: C=3,R=21,V=0.01
2009-11-15 12:41:44,278 [Thread-19] INFO: solve: :OM: C=3,R=22,V=0.22
2009-11-15 12:41:44,278 [Thread-19] INFO: solve: :OM: C=4,R=1,V=0.12
2009-11-15 12:41:44,279 [Thread-19] INFO: solve: :OM: C=4,R=2,V=0.41
2009-11-15 12:41:44,279 [Thread-19] INFO: solve: :OM: C=4,R=3,V=0.52
2009-11-15 12:41:44,279 [Thread-19] INFO: solve: :OM: C=4,R=4,V=0.17
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=5,V=0.23
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=6,V=0.23
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=7,V=0.66
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=8,V=0.39
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=9,V=0.53
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=10,V=0.70
2009-11-15 12:41:44,280 [Thread-19] INFO: solve: :OM: C=4,R=11,V=0.43
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=12,V=0.66
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=13,V=0.15
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=14,V=0.57
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=15,V=0.42
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=16,V=0.54
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=17,V=0.22
2009-11-15 12:41:44,281 [Thread-19] INFO: solve: :OM: C=4,R=18,V=0.50
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=4,R=19,V=0.66
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=4,R=20,V=0.24
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=4,R=21,V=0.01
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=4,R=22,V=0.22
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=5,R=1,V=0.12
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=5,R=2,V=0.41
2009-11-15 12:41:44,282 [Thread-19] INFO: solve: :OM: C=5,R=3,V=0.52
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=4,V=0.17
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=5,V=0.23
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=6,V=0.23
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=7,V=0.66
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=8,V=0.39
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=9,V=0.53
2009-11-15 12:41:44,283 [Thread-19] INFO: solve: :OM: C=5,R=10,V=0.70
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=11,V=0.43
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=12,V=0.66
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=13,V=0.15
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=14,V=0.57
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=15,V=0.42
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=16,V=0.54
2009-11-15 12:41:44,284 [Thread-19] INFO: solve: :OM: C=5,R=17,V=0.22
```

```
2009-11-15 12:41:44,285 [Thread-19] INFO: solve: :OM: C=5,R=18,V=0.50
2009-11-15 12:41:44,285 [Thread-19] INFO: solve: :OM: C=5,R=19,V=0.66
2009-11-15 12:41:44,285 [Thread-19] INFO: solve: :OM: C=5,R=20,V=0.24
2009-11-15 12:41:44,285 [Thread-19] INFO: solve: :OM: C=5,R=21,V=0.01
2009-11-15 12:41:44,285 [Thread-19] INFO: solve: :OM: C=5,R=22,V=0.22
2009-11-15 12:41:44,286 [Thread-19] INFO: solve: :OM: C=6,R=1,V=0.12
2009-11-15 12:41:44,286 [Thread-19] INFO: solve: :OM: C=6,R=2,V=0.41
2009-11-15 12:41:44,286 [Thread-19] INFO: solve: :OM: C=6,R=3,V=0.52
2009-11-15 12:41:44,286 [Thread-19] INFO: solve: :OM: C=6,R=4,V=0.17
2009-11-15 12:41:44,286 [Thread-19] INFO: solve: :OM: C=6,R=5,V=0.23
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=6,V=0.23
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=7,V=0.66
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=8,V=0.39
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=9,V=0.53
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=10,V=0.70
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=11,V=0.43
2009-11-15 12:41:44,287 [Thread-19] INFO: solve: :OM: C=6,R=12,V=0.66
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=13,V=0.15
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=14,V=0.57
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=15,V=0.42
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=16,V=0.54
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=17,V=0.22
2009-11-15 12:41:44,288 [Thread-19] INFO: solve: :OM: C=6,R=18,V=0.50
2009-11-15 12:41:44,289 [Thread-19] INFO: solve: :OM: C=6,R=19,V=0.66
2009-11-15 12:41:44,289 [Thread-19] INFO: solve: :OM: C=6,R=20,V=0.24
2009-11-15 12:41:44,289 [Thread-19] INFO: solve: :OM: C=6,R=21,V=0.01
2009-11-15 12:41:44,289 [Thread-19] INFO: solve: :OM: C=6,R=22,V=0.22
2009-11-15 12:41:44,290 [Thread-19] INFO: solve: :OM: C=7,R=1,V=0.12
2009-11-15 12:41:44,290 [Thread-19] INFO: solve: :OM: C=7,R=2,V=0.41
2009-11-15 12:41:44,290 [Thread-19] INFO: solve: :OM: C=7,R=3,V=0.52
2009-11-15 12:41:44,290 [Thread-19] INFO: solve: :OM: C=7,R=4,V=0.17
2009-11-15 12:41:44,290 [Thread-19] INFO: solve: :OM: C=7,R=5,V=0.23
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=6,V=0.23
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=7,V=0.66
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=8,V=0.39
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=9,V=0.53
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=10,V=0.70
2009-11-15 12:41:44,291 [Thread-19] INFO: solve: :OM: C=7,R=11,V=0.43
2009-11-15 12:41:44,292 [Thread-19] INFO: solve: :OM: C=7,R=12,V=0.66
2009-11-15 12:41:44,292 [Thread-19] INFO: solve: :OM: C=7,R=13,V=0.15
2009-11-15 12:41:44,292 [Thread-19] INFO: solve: :OM: C=7,R=14,V=0.57
2009-11-15 12:41:44,292 [Thread-19] INFO: solve: :OM: C=7,R=15,V=0.43
2009-11-15 12:41:44,293 [Thread-19] INFO: solve: :OM: C=7,R=16,V=0.54
2009-11-15 12:41:44,293 [Thread-19] INFO: solve: :OM: C=7,R=17,V=0.22
```

```
2009-11-15 12:41:44,293 [Thread-19] INFO: solve: :OM: C=7,R=18,V=0.50
2009-11-15 12:41:44,294 [Thread-19] INFO: solve: :OM: C=7,R=19,V=0.66
2009-11-15 12:41:44,294 [Thread-19] INFO: solve: :OM: C=7,R=20,V=0.24
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=7,R=21,V=0.01
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=7,R=22,V=0.22
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=8,R=1,V=0.12
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=8,R=2,V=0.41
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=8,R=3,V=0.52
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=8,R=4,V=0.17
2009-11-15 12:41:44,295 [Thread-19] INFO: solve: :OM: C=8,R=5,V=0.23
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=6,V=0.23
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=7,V=0.66
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=8,V=0.39
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=9,V=0.53
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=10,V=0.70
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=11,V=0.43
2009-11-15 12:41:44,296 [Thread-19] INFO: solve: :OM: C=8,R=12,V=0.66
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=13,V=0.15
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=14,V=0.57
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=15,V=0.42
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=16,V=0.54
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=17,V=0.22
2009-11-15 12:41:44,297 [Thread-19] INFO: solve: :OM: C=8,R=18,V=0.50
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=8,R=19,V=0.66
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=8,R=20,V=0.24
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=8,R=21,V=0.01
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=8,R=22,V=0.22
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=9,R=1,V=0.12
2009-11-15 12:41:44,298 [Thread-19] INFO: solve: :OM: C=9,R=2,V=0.41
2009-11-15 12:41:44,299 [Thread-19] INFO: solve: :OM: C=9,R=3,V=0.52
2009-11-15 12:41:44,299 [Thread-19] INFO: solve: :OM: C=9,R=4,V=0.17
2009-11-15 12:41:44,299 [Thread-19] INFO: solve: :OM: C=9,R=5,V=0.23
2009-11-15 12:41:44,299 [Thread-19] INFO: solve: :OM: C=9,R=6,V=0.24
2009-11-15 12:41:44,300 [Thread-19] INFO: solve: :OM: C=9,R=7,V=0.66
2009-11-15 12:41:44,300 [Thread-19] INFO: solve: :OM: C=9,R=8,V=0.39
2009-11-15 12:41:44,300 [Thread-19] INFO: solve: :OM: C=9,R=9,V=0.53
2009-11-15 12:41:44,300 [Thread-19] INFO: solve: :OM: C=9,R=10,V=0.70
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=11,V=0.43
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=12,V=0.66
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=13,V=0.15
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=14,V=0.57
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=15,V=0.42
2009-11-15 12:41:44,302 [Thread-19] INFO: solve: :OM: C=9,R=16,V=0.54
2009-11-15 12:41:44,303 [Thread-19] INFO: solve: :OM: C=9,R=17,V=0.22
```

```
2009-11-15 12:41:44,303 [Thread-19] INFO: solve: :OM: C=9,R=18,V=0.50
2009-11-15 12:41:44,303 [Thread-19] INFO: solve: :OM: C=9,R=19,V=0.66
2009-11-15 12:41:44,304 [Thread-19] INFO: solve: :OM: C=9,R=20,V=0.24
2009-11-15 12:41:44,304 [Thread-19] INFO: solve: :OM: C=9,R=21,V=0.01
2009-11-15 12:41:44,304 [Thread-19] INFO: solve: :OM: C=9,R=22,V=0.22
2009-11-15 12:41:44,304 [Thread-19] INFO: solve: :OM: C=10,R=1,V=0.12
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=2,V=0.41
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=3,V=0.52
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=4,V=0.17
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=5,V=0.23
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=6,V=0.23
2009-11-15 12:41:44,305 [Thread-19] INFO: solve: :OM: C=10,R=7,V=0.66
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=8,V=0.39
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=9,V=0.53
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=10,V=0.70
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=11,V=0.43
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=12,V=0.66
2009-11-15 12:41:44,306 [Thread-19] INFO: solve: :OM: C=10,R=13,V=0.15
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=14,V=0.57
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=15,V=0.42
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=16,V=0.54
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=17,V=0.22
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=18,V=0.50
2009-11-15 12:41:44,307 [Thread-19] INFO: solve: :OM: C=10,R=19,V=0.66
2009-11-15 12:41:44,308 [Thread-19] INFO: solve: :OM: C=10,R=20,V=0.24
2009-11-15 12:41:44,308 [Thread-19] INFO: solve: :OM: C=10,R=21,V=0.01
2009-11-15 12:41:44,308 [Thread-19] INFO: solve: :OM: C=10,R=22,V=0.22
2009-11-15 12:41:44,309 [Thread-19] INFO: run:
:CCP: (1.4199185.26214498) Received Opinion tuple from OM.
[O=1;R=19,V=0.66;R=18,V=0.5;R=17,V=0.22;R=16,V=0.54;
R=15,V=0.42;R=14,V=0.57;R=13,V=0.15;R=12,V=0.66;
R=11,V=0.43;R=10,V=0.7;R=9,V=0.53;R=8,V=0.39;
R=7,V=0.66;R=6,V=0.23;R=5,V=0.23;
R=4,V=0.17;R=3,V=0.52;R=2,V=0.41;R=1,V=0.12;
R=22,V=0.22;R=21,V=0.01;R=20,V=0.24]
2009-11-15 12:41:44,310 [Thread-19] INFO: put: :OA:
Inserting Opinion Tuple into OA. (O=1,W=0.9)
2009-11-15 12:41:44,310 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=19,VxW=0.59]
2009-11-15 12:41:44,310 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=18,VxW=0.45]
2009-11-15 12:41:44,310 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=17,VxW=0.2]
2009-11-15 12:41:44,311 [Thread-19] INFO: put: :OA:
```

```
Inserting values into OA. [R=16,VxW=0.49]
2009-11-15 12:41:44,311 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=15,VxW=0.38]
2009-11-15 12:41:44,311 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=14,VxW=0.51]
2009-11-15 12:41:44,312 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=13,VxW=0.14]
2009-11-15 12:41:44,312 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=12,VxW=0.59]
2009-11-15 12:41:44,312 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=11,VxW=0.39]
2009-11-15 12:41:44,313 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=10,VxW=0.63]
2009-11-15 12:41:44,313 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=9,VxW=0.48]
2009-11-15 12:41:44,313 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=8,VxW=0.35]
2009-11-15 12:41:44,314 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=7,VxW=0.59]
2009-11-15 12:41:44,314 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=6,VxW=0.21]
2009-11-15 12:41:44,315 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=5,VxW=0.21]
2009-11-15 12:41:44,315 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=4,VxW=0.15]
2009-11-15 12:41:44,315 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=3,VxW=0.47]
2009-11-15 12:41:44,316 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=2,VxW=0.37]
2009-11-15 12:41:44,316 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=1,VxW=0.11]
2009-11-15 12:41:44,316 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=22,VxW=0.2]
2009-11-15 12:41:44,316 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=21,VxW=0.01]
2009-11-15 12:41:44,317 [Thread-19] INFO: put: :OA:
Inserting values into OA. [R=20,VxW=0.22]
2009-11-15 12:41:44,317 [Thread-9] INFO: <init>: :CP:
(1.4199185.6570132) Initialized Child Correlation Process.
2009-11-15 12:41:44,318 [Thread-20] INFO: run: :CCP:
(1.4199185.6570132) Forked Child CP. [O=2]
2009-11-15 12:41:44,318 [Thread-20] INFO: getOpinionMatrix:
:CMP: Contacted Customized Metrics Pool to return OM for opinion: 2
2009-11-15 12:41:44,318 [Thread-20] INFO: solve: :OM:
```

```
Generating OM=CALL OPINION_MATRIX_SELECT(2, NULL, 0);
2009-11-15 12:53:42,947 [Thread-20] INFO: solve: :OM: C=1,R=1,V=0.99
2009-11-15 12:53:42,947 [Thread-20] INFO: solve: :OM: C=1,R=2,V=0.99
2009-11-15 12:53:42,947 [Thread-20] INFO: solve: :OM: C=1,R=3,V=0.99
2009-11-15 12:53:42,947 [Thread-20] INFO: solve: :OM: C=1,R=4,V=0.99
2009-11-15 12:53:42,947 [Thread-20] INFO: solve: :OM: C=1,R=5,V=0.99
2009-11-15 12:53:42,948 [Thread-20] INFO: solve: :OM: C=1,R=6,V=0.99
2009-11-15 12:53:42,948 [Thread-20] INFO: solve: :OM: C=1,R=7,V=0.99
2009-11-15 12:53:42,948 [Thread-20] INFO: solve: :OM: C=1,R=8,V=0.99
2009-11-15 12:53:42,948 [Thread-20] INFO: solve: :OM: C=1,R=9,V=0.99
2009-11-15 12:53:42,952 [Thread-20] INFO: solve: :OM: C=1,R=10,V=0.99
2009-11-15 12:53:42,952 [Thread-20] INFO: solve: :OM: C=1,R=11,V=0.99
2009-11-15 12:53:42,953 [Thread-20] INFO: solve: :OM: C=1,R=12,V=0.99
2009-11-15 12:53:42,953 [Thread-20] INFO: solve: :OM: C=1,R=13,V=0.99
2009-11-15 12:53:42,953 [Thread-20] INFO: solve: :OM: C=1,R=14,V=0.99
2009-11-15 12:53:42,954 [Thread-20] INFO: solve: :OM: C=1,R=15,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=16,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=17,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=18,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=19,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=20,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=21,V=0.99
2009-11-15 12:53:42,955 [Thread-20] INFO: solve: :OM: C=1,R=22,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=1,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=2,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=3,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=4,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=5,V=0.99
2009-11-15 12:53:42,956 [Thread-20] INFO: solve: :OM: C=2,R=6,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=7,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=8,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=9,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=10,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=11,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=12,V=0.99
2009-11-15 12:53:42,957 [Thread-20] INFO: solve: :OM: C=2,R=13,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=14,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=15,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=16,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=17,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=18,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=19,V=0.99
2009-11-15 12:53:42,958 [Thread-20] INFO: solve: :OM: C=2,R=20,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=2,R=21,V=0.99
```

```
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=2,R=22,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=3,R=1,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=3,R=2,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=3,R=3,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=3,R=4,V=0.99
2009-11-15 12:53:42,959 [Thread-20] INFO: solve: :OM: C=3,R=5,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=6,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=7,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=8,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=9,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=10,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=11,V=0.99
2009-11-15 12:53:42,960 [Thread-20] INFO: solve: :OM: C=3,R=12,V=0.99
2009-11-15 12:53:42,961 [Thread-20] INFO: solve: :OM: C=3,R=13,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=14,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=15,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=16,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=17,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=18,V=0.99
2009-11-15 12:53:42,962 [Thread-20] INFO: solve: :OM: C=3,R=19,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=3,R=20,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=3,R=21,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=3,R=22,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=4,R=1,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=4,R=2,V=0.99
2009-11-15 12:53:42,963 [Thread-20] INFO: solve: :OM: C=4,R=3,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=4,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=5,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=6,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=7,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=8,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=9,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=10,V=0.99
2009-11-15 12:53:42,964 [Thread-20] INFO: solve: :OM: C=4,R=11,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=12,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=13,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=14,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=15,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=16,V=0.99
2009-11-15 12:53:42,965 [Thread-20] INFO: solve: :OM: C=4,R=17,V=0.99
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=4,R=18,V=0.99
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=4,R=19,V=0.99
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=4,R=20,V=0.99
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=4,R=21,V=0.99
```

```
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=4,R=22,V=0.99
2009-11-15 12:53:42,966 [Thread-20] INFO: solve: :OM: C=5,R=1,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=2,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=3,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=4,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=5,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=6,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=7,V=0.99
2009-11-15 12:53:42,967 [Thread-20] INFO: solve: :OM: C=5,R=8,V=0.99
2009-11-15 12:53:42,968 [Thread-20] INFO: solve: :OM: C=5,R=9,V=0.99
2009-11-15 12:53:42,968 [Thread-20] INFO: solve: :OM: C=5,R=10,V=0.99
2009-11-15 12:53:42,968 [Thread-20] INFO: solve: :OM: C=5,R=11,V=0.99
2009-11-15 12:53:42,968 [Thread-20] INFO: solve: :OM: C=5,R=12,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=13,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=14,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=15,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=16,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=17,V=0.99
2009-11-15 12:53:42,969 [Thread-20] INFO: solve: :OM: C=5,R=18,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=5,R=19,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=5,R=20,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=5,R=21,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=5,R=22,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=6,R=1,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=6,R=2,V=0.99
2009-11-15 12:53:42,970 [Thread-20] INFO: solve: :OM: C=6,R=3,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=4,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=5,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=6,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=7,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=8,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=9,V=0.99
2009-11-15 12:53:42,971 [Thread-20] INFO: solve: :OM: C=6,R=10,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=11,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=12,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=13,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=14,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=15,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=16,V=0.99
2009-11-15 12:53:42,972 [Thread-20] INFO: solve: :OM: C=6,R=17,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=6,R=18,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=6,R=19,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=6,R=20,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=6,R=21,V=0.99
```

```
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=6,R=22,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=7,R=1,V=0.99
2009-11-15 12:53:42,973 [Thread-20] INFO: solve: :OM: C=7,R=2,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=3,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=4,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=5,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=6,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=7,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=8,V=0.99
2009-11-15 12:53:42,974 [Thread-20] INFO: solve: :OM: C=7,R=9,V=0.99
2009-11-15 12:53:42,975 [Thread-20] INFO: solve: :OM: C=7,R=10,V=0.99
2009-11-15 12:53:42,975 [Thread-20] INFO: solve: :OM: C=7,R=11,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=12,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=13,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=14,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=15,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=16,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=17,V=0.99
2009-11-15 12:53:42,976 [Thread-20] INFO: solve: :OM: C=7,R=18,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=7,R=19,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=7,R=20,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=7,R=21,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=7,R=22,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=8,R=1,V=0.99
2009-11-15 12:53:42,977 [Thread-20] INFO: solve: :OM: C=8,R=2,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=3,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=4,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=5,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=6,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=7,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=8,V=0.99
2009-11-15 12:53:42,978 [Thread-20] INFO: solve: :OM: C=8,R=9,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=10,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=11,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=12,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=13,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=14,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=15,V=0.99
2009-11-15 12:53:42,979 [Thread-20] INFO: solve: :OM: C=8,R=16,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=17,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=18,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=19,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=20,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=21,V=0.99
```

```
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=8,R=22,V=0.99
2009-11-15 12:53:42,980 [Thread-20] INFO: solve: :OM: C=9,R=1,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=2,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=3,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=4,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=5,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=6,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=7,V=0.99
2009-11-15 12:53:42,981 [Thread-20] INFO: solve: :OM: C=9,R=8,V=0.99
2009-11-15 12:53:42,982 [Thread-20] INFO: solve: :OM: C=9,R=9,V=0.99
2009-11-15 12:53:42,982 [Thread-20] INFO: solve: :OM: C=9,R=10,V=0.99
2009-11-15 12:53:42,982 [Thread-20] INFO: solve: :OM: C=9,R=11,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=12,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=13,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=14,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=15,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=16,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=17,V=0.99
2009-11-15 12:53:42,983 [Thread-20] INFO: solve: :OM: C=9,R=18,V=0.99
2009-11-15 12:53:42,984 [Thread-20] INFO: solve: :OM: C=9,R=19,V=0.99
2009-11-15 12:53:42,989 [Thread-20] INFO: solve: :OM: C=9,R=20,V=0.99
2009-11-15 12:53:42,989 [Thread-20] INFO: solve: :OM: C=9,R=21,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=9,R=22,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=1,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=2,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=3,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=4,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=5,V=0.99
2009-11-15 12:53:42,990 [Thread-20] INFO: solve: :OM: C=10,R=6,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=7,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=8,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=9,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=10,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=11,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=12,V=0.99
2009-11-15 12:53:42,991 [Thread-20] INFO: solve: :OM: C=10,R=13,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=14,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=15,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=16,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=17,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=18,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=19,V=0.99
2009-11-15 12:53:42,992 [Thread-20] INFO: solve: :OM: C=10,R=20,V=0.99
2009-11-15 12:53:42,993 [Thread-20] INFO: solve: :OM: C=10,R=21,V=0.99
```

```
2009-11-15 12:53:42,993 [Thread-20] INFO: solve: :OM: C=10,R=22,V=0.99
2009-11-15 12:53:42,994 [Thread-20] INFO: run:
:CCP: (1.4199185.6570132) Received Opinion tuple from OM.
[O=2;R=19,V=0.99;R=18,V=0.99;R=17,V=0.99;
R=16,V=0.99;R=15,V=0.99;R=14,V=0.99;R=13,V=0.99;
R=12,V=0.99;R=11,V=0.99;R=10,V=0.99;R=9,V=0.99;
R=8,V=0.99;R=7,V=0.99;R=6,V=0.99;R=5,V=0.99;
R=4,V=0.99;R=3,V=0.99;R=2,V=0.99;R=1,V=0.99;
R=22,V=0.99;R=21,V=0.99;R=20,V=0.99]
2009-11-15 12:53:42,994 [Thread-20] INFO: put: :OA:
Inserting Opinion Tuple into OA. (O=2,W=0.1)
2009-11-15 12:53:42,995 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=19,VxW=0.1]
2009-11-15 12:53:42,995 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=18,VxW=0.1]
2009-11-15 12:53:42,995 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=17,VxW=0.1]
2009-11-15 12:53:42,996 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=16,VxW=0.1]
2009-11-15 12:53:42,997 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=15,VxW=0.1]
2009-11-15 12:53:42,997 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=14,VxW=0.1]
2009-11-15 12:53:42,997 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=13,VxW=0.1]
2009-11-15 12:53:42,998 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=12,VxW=0.1]
2009-11-15 12:53:42,998 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=11,VxW=0.1]
2009-11-15 12:53:42,998 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=10,VxW=0.1]
2009-11-15 12:53:43,001 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=9,VxW=0.1]
2009-11-15 12:53:43,002 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=8,VxW=0.1]
2009-11-15 12:53:43,002 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=7,VxW=0.1]
2009-11-15 12:53:43,002 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=6,VxW=0.1]
2009-11-15 12:53:43,002 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=5,VxW=0.1]
2009-11-15 12:53:43,003 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=4,VxW=0.1]
2009-11-15 12:53:43,003 [Thread-20] INFO: put: :OA:
```

```
Inserting values into OA. [R=3,VxW=0.1]
2009-11-15 12:53:43,003 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=2,VxW=0.1]
2009-11-15 12:53:43,004 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=1,VxW=0.1]
2009-11-15 12:53:43,005 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=22,VxW=0.1]
2009-11-15 12:53:43,005 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=21,VxW=0.1]
2009-11-15 12:53:43,005 [Thread-20] INFO: put: :OA:
Inserting values into OA. [R=20,VxW=0.1]
2009-11-15 12:53:43,006 [Thread-9] INFO: process:
:CP: (1.4199185) All Child CP forks returned. Performing summary on OA.
2009-11-15 12:53:43,282 [Thread-9] INFO: process:
:RA: Initialized TDR Engine. contextValue=0.45
2009-11-15 12:53:43,283 [Thread-9] INFO: process:
:RA: Generated Resource Value Table

2009-11-15 12:53:43,283 [Thread-9] INFO: process: :RA: R=19,V=0.69
2009-11-15 12:53:43,307 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,307 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.6

2009-11-15 12:53:43,307 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,309 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,309 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.25
2009-11-15 12:53:43,309 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,309 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,310 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,310 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,310 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,310 [Thread-9] INFO: process: :RA: R=18,V=0.55
2009-11-15 12:53:43,311 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,312 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.55

2009-11-15 12:53:43,312 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,312 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,312 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.1
2009-11-15 12:53:43,313 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,313 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,313 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,314 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,314 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
```

```
2009-11-15 12:53:43,314 [Thread-9] INFO: process: :RA: R=17,V=0.3
2009-11-15 12:53:43,315 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,315 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,315 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,316 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,316 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,316 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,316 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,317 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,317 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,317 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,317 [Thread-9] INFO: process: :RA: R=16,V=0.59
2009-11-15 12:53:43,318 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,318 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.58

2009-11-15 12:53:43,318 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,319 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,320 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.18
2009-11-15 12:53:43,320 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,320 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,320 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,320 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,321 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,321 [Thread-9] INFO: process: :RA: R=15,V=0.48
2009-11-15 12:53:43,322 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,322 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.48

2009-11-15 12:53:43,322 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,322 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,323 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,323 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,323 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.04
2009-11-15 12:53:43,323 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,323 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,324 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.04
2009-11-15 12:53:43,324 [Thread-9] INFO: process: :RA: R=14,V=0.61
2009-11-15 12:53:43,325 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,325 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.59
```

```
2009-11-15 12:53:43,325 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,325 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.22
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,326 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,327 [Thread-9] INFO: process: :RA: R=13,V=0.24
2009-11-15 12:53:43,328 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,328 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,328 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,328 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,328 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,329 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,329 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,330 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,330 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,330 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,330 [Thread-9] INFO: process: :RA: R=12,V=0.69
2009-11-15 12:53:43,331 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,331 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.6

2009-11-15 12:53:43,332 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,332 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,332 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.25
2009-11-15 12:53:43,332 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,332 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,333 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,333 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,333 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,333 [Thread-9] INFO: process: :RA: R=11,V=0.49
2009-11-15 12:53:43,334 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,334 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.49

2009-11-15 12:53:43,334 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,335 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,335 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,335 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,335 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.02
```

```
2009-11-15 12:53:43,335 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,336 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,336 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.02
2009-11-15 12:53:43,336 [Thread-9] INFO: process: :RA: R=10,V=0.73
2009-11-15 12:53:43,337 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,337 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.6

2009-11-15 12:53:43,338 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,338 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,338 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.25
2009-11-15 12:53:43,339 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,339 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,339 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,339 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,340 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,340 [Thread-9] INFO: process: :RA: R=9,V=0.58
2009-11-15 12:53:43,341 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,341 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.57

2009-11-15 12:53:43,341 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,341 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,342 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.16
2009-11-15 12:53:43,342 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,342 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,342 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,342 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,343 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,343 [Thread-9] INFO: process: :RA: R=8,V=0.45
2009-11-15 12:53:43,344 [Thread-9] INFO: process:
:DRE: Rule=9,Degree=0.0

2009-11-15 12:53:43,344 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.45
2009-11-15 12:53:43,344 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,344 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.1
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,345 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,346 [Thread-9] INFO: process: :RA: R=7,V=0.69
```

```
2009-11-15 12:53:43,347 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,347 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.6

2009-11-15 12:53:43,347 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,347 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,347 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.25
2009-11-15 12:53:43,348 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,348 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,348 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,349 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,349 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,349 [Thread-9] INFO: process: :RA: R=6,V=0.31
2009-11-15 12:53:43,350 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,350 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,350 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,351 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,352 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,352 [Thread-9] INFO: process: :RA: R=5,V=0.31
2009-11-15 12:53:43,353 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,353 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,353 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,353 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,354 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,355 [Thread-9] INFO: process: :RA: R=4,V=0.25
2009-11-15 12:53:43,355 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,356 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,356 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
```

```
2009-11-15 12:53:43,356 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,356 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,356 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,357 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,357 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,358 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,358 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,358 [Thread-9] INFO: process: :RA: R=3,V=0.57
2009-11-15 12:53:43,359 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,359 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.57

2009-11-15 12:53:43,359 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,359 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,360 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.14
2009-11-15 12:53:43,360 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,360 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.0
2009-11-15 12:53:43,360 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.1
2009-11-15 12:53:43,360 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,361 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.0
2009-11-15 12:53:43,361 [Thread-9] INFO: process: :RA: R=2,V=0.47
2009-11-15 12:53:43,362 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,362 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.47

2009-11-15 12:53:43,362 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,362 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,362 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.06
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.06
2009-11-15 12:53:43,363 [Thread-9] INFO: process: :RA: R=22,V=0.3
2009-11-15 12:53:43,364 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,365 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,365 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,365 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,365 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,365 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,365 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,366 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
```

```
2009-11-15 12:53:43,366 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,366 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,367 [Thread-9] INFO: process: :RA: R=1,V=0.21
2009-11-15 12:53:43,367 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,368 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,368 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,368 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,368 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,368 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,369 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,369 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,369 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,369 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,369 [Thread-9] INFO: process: :RA: R=21,V=0.11
2009-11-15 12:53:43,370 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,370 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.39

2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.22
2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,371 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,372 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,372 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,372 [Thread-9] INFO: process: :RA: R=20,V=0.32
2009-11-15 12:53:43,373 [Thread-9] INFO: process: :DRE: Rule=9,Degree=0.0
2009-11-15 12:53:43,373 [Thread-9] INFO: process:
:DRE: Rule=trust_level,Degree=0.4

2009-11-15 12:53:43,373 [Thread-9] INFO: process: :DRE: Rule=8,Degree=0.0
2009-11-15 12:53:43,373 [Thread-9] INFO: process: :DRE: Rule=7,Degree=0.0
2009-11-15 12:53:43,374 [Thread-9] INFO: process: :DRE: Rule=6,Degree=0.0
2009-11-15 12:53:43,374 [Thread-9] INFO: process: :DRE: Rule=5,Degree=0.25
2009-11-15 12:53:43,374 [Thread-9] INFO: process: :DRE: Rule=4,Degree=0.25
2009-11-15 12:53:43,374 [Thread-9] INFO: process: :DRE: Rule=3,Degree=0.0
2009-11-15 12:53:43,374 [Thread-9] INFO: process: :DRE: Rule=2,Degree=0.1
2009-11-15 12:53:43,375 [Thread-9] INFO: process: :DRE: Rule=1,Degree=0.1
2009-11-15 12:53:43,415 [Thread-9] INFO: process: :RA: Report generated.
```

.3 GREPTrust Testbed Diagrams, Simulation Algorithms & Outputs

.3.1 Diagrams

Figure 9 illustrates the different packages comprising of the GREPTrust testbed. The most predominant package is GREPTrustAnalysis. This package contains the main testbed simulation algorithms aggregated under a single Python script (exp_testbed_experiment.py) which generates the feedback data and uploads it to GREPTrustDB, runs the trust evaluation to generate the trust comparison report, runs the jobs simulation and comparison analysis and finally outputs the comparison results and generates a scenario trend. The GREPTrustAnalysis package depends on the GREPTrustAdmin package for loading the GREPTrustDB data store with reputation feedback data, the GREPTrustTestbed package for performing resource evaluation, the SADB package for accessing the SEDOL analytics data during the testbed simulation and the Matplolib Python library for generating the comparison graphs and scenario trends. The GREPTrustTestbed package itself depends on the GridSim environment and GREPTrust and GREPTrustAggregator packages.

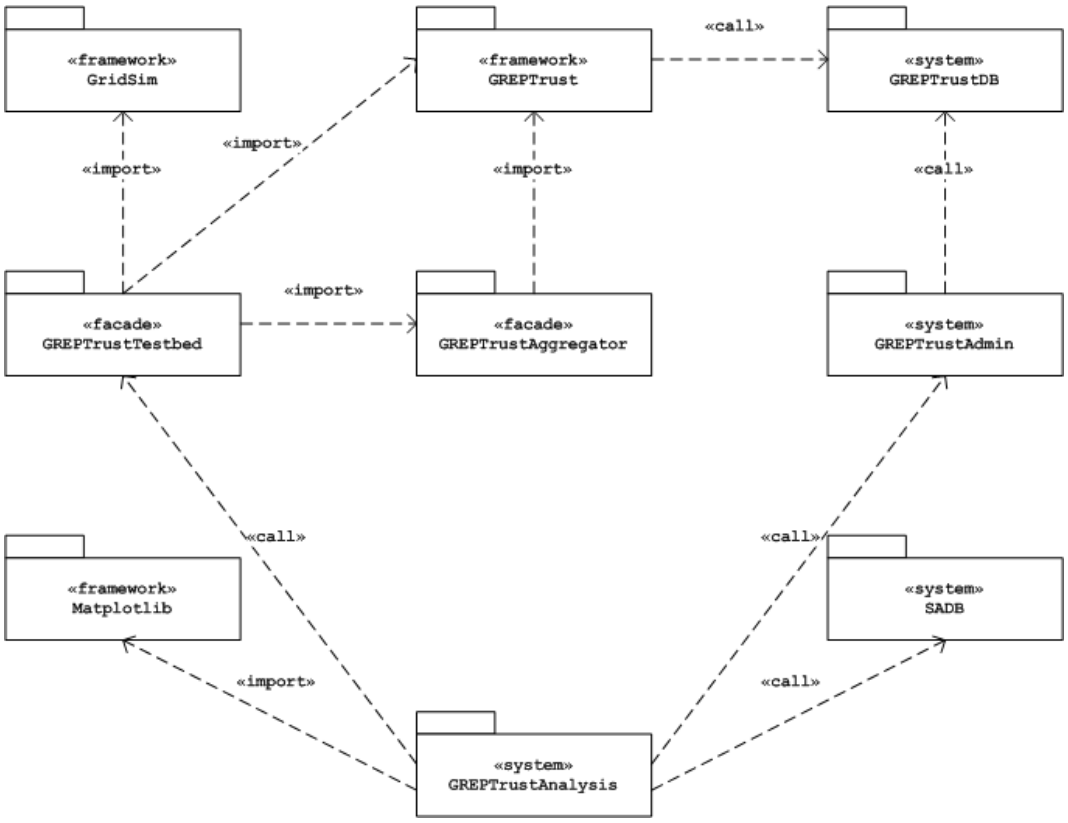


Figure 9: GREPTrust testbed package diagram

.3.2 Simulation Algorithms

The following collection of algorithms constitutes the simulation process which was elaborately described in Chapter 5. Algorithms 11 and 12 execute the reliability and availability volume distribution corruption functions respectively. In Algorithm 11 - Resource::DoReliability(VD) line 7, a modified volume distribution VD' is returned after randomising the bin values of the archetype volume distribution VD. In Algorithm 12 - DoAvailability(VD) line 10 a modified volume distribution VD' is returned after setting all its bin values to -1 (lines 7 - 9). Algorithms 11 and 12 are invoked by algorithm 16 - Resource::Compute(S, VD) which makes the decision on which fail factor to execute utilising algorithm 13 - Resource::GetFactor().

In algorithm 13 line 7, a random integer $Rand$ stores a value between 1 and 100 inclusive. Each of the fail factors in the factor matrix $FMat$ is compared against $Rand$ - if the fail value is less than $Rand$ (line 9) that it is added to an intermediate matrix $FMat'$. The permissible size of $FMat'$ therefore is $0 \leq FMat' \leq 2$. If the size of $FMat'$ is 0 then no fail factors are returned (line 15); otherwise if the size is greater than 1 then a secondary randomisation occurs which returns either availability or reliability fail factors (line 19). Finally, if the size is exactly 1 then the stored factor ID and value are returned (line 21). The modified volume distribution VD' is returned in algorithm 16, line 17 and stored into the destined table T in algorithm 15, line 7.

Algorithm 14 - GridJob::Run() simulates a running Grid job comprising of N running processes $Procs$. Each process is associated with an ID, a computing resource (such as a cluster) and scheduled a list of SEDOL identifiers to process. The scheduling is performed by algorithm 17 - GridJob::Schedule(), line 36. Finally, algorithm 18 - GridJobExecutionManager::Execute(RP, T) wraps the entire job simulation process. In lines 13 - 14, two lists are created - P1RS and P2RS comprising of GridPP and GREPTrust recommended resources respectively. In line 19, two Grid jobs are created ; one for GridPP and the other - for GREPTrust. Each Grid job is allocated with an ID a list of SEDOL identifiers and the resource list.

Algorithm 11: Resource::DoReliability(VD) algorithm

1: VD: archetype volume distribution
2: VD': modified volume distribution
3: BIDs: array of bin identifiers
4: BVals: array of bin values
5: $BIDs \leftarrow VD[Column(0)]$;
6: $BVals \leftarrow VD[Column(1)]$;
7: $VD' \leftarrow Pack(BIDs, Randomize(BVals))$;
8: **return** VD';

Algorithm 12: Resource::DoAvailability(VD) algorithm

```
1: VD: archetype volume distribution
2: VD': modified volume distribution
3: BIDs: array of bin identifiers
4: BVals: array of bin values
5: BIDs ← VD[Column(0)];
6: BVals ← VD[Column(1)];
7: for all Val in BVals do
8:    Val ← −1;
9: end for
10: VD′ ← Pack(BIDs, BVals);
11: return VD′;
```

Algorithm 13: Resource::GetFactor() algorithm

```
1: Rand: random integer between 1 and 100
2: FMat: factors matrix
3: FMat': intermediate matrix
4: FID: factor identifier
5: FVal: factor value
6: Size: cardinality of the intermediate matrix
7: Rand ← Random(1, 100);
8: for all FID, FVal in FMat do
9:    if FVal × 100 < Rand then
10:      FMat′ ← Add(FID, FVal);
11:   end if
12: end for
13: Size ← Length(FMat′);
14: if Size = 0 then
15:   return 0;
16: end if
17: if Size > 1 then
18:   Rand ← Random(0, Size − 1);
19:   FID, FVal ← FMat′[Rand];
20: else
21:   FID, FVal ← FMat′[0];
22: end if
23: return Pair(FID, FVal);
```

Algorithm 14: GridJob::Run() algorithm

```
1: Start: start time
2: End: end time
3: P: individual process
4: Procs: array of processes
5: Start ← Clock();
6: for all P in Procs do
7:    Run(P);
8: end for
9: End ← Clock();
10: print End − Start
```

Algorithm 15: Process::Run() algorithm

```
1: SIDs: SEDOL identifiers array allocated for the running process
2: VD': modified volume distribution
3: T: target table for saving the modified volume distribution
4: for all S in SIDs do
5:    VD' ← Compute(S, Load(S, 'voldist'));
6:    Table ← Procs[Name];
7:    Save(S, VD', T);
8: end for
```

Algorithm 16: Resource::Compute(S, VD) algorithm

```
1: S: SEDOL identifier
2: VD: archetype volume distribution
3: VD': modified volume distribution
4: FPair: factor pair (FID, FVal)
5: FID: factor identifier
6: FVal: factor value
7: FPair ← GetFactor();
8: if FPair = null then
9:    return VD;
10: end if
11: FID, FVal ← FPair;
12: if FID = 1 then
13:    VD' ← DoAvailability(VD);
14: else if FID = 2 then
15:    VD' ← DoReliability(VD);
16: end if
17: return VD';
```

Algorithm 17: GridJob::Schedule() algorithm

```
1: R: resource object
2: RID: resource object identifier
3: SL: array of SEDOL identifiers
4: RS: array of Resource objects
5: NSedols: size of SEDOL array
6: NResources: size of resources array
7: NAllocate: quotient of SEDOL identifiers
8: NRemainder: remainder of SEDOL identifiers
9: Start: start position
10: Stop: stop position
11: Step: step size
12: Index: count index
13: List: sliced array of SEDOL identifiers
14: Plist: array of processes
15: Name: name of the calling Grid job

16: NSedols ← Size(SL);
17: NResources ← Size(RS);
18: NAllocate,NRemainder ← Divmod(NSedols, NResources);

19: Start ← 0;
20: Stop ← NAllocate;
21: Step ← 1;
22: Index ← 1;
23: Plist ← [];

24: if NSedols < NResources then
25:    Append(Plist, Name, Resources[0], SL);
26:    return Plist;
27: end if

28: for all RinNResources do
29:    if RID = ID(resources[−1]) then
30:       Stop ← Stop + NRemainder;
31:    end if

32:    Index ← Index + 1;
33:    List ← slice(SL, Start, Stop, Step);
34:    Start ← Stop;
35:    Stop ← NAllocate × Index;

36:    Plist → Append(Name, R, List);
37: end for

38: return Plist;
```

Algorithm 18: GridJobExecutionManager::Execute(RP, T) algorithm

```
1: TCR: trust comparison report file system path
2: T: threshold used for selecting the final array of resources
3: RMatrix: report matrix
4: SIDs: array of SEDOL identifiers
5: RS: array of resource objects
6: GridJobs: array of Grid jobs
7: P1RS: array of resource objects allocated to process1
8: P2RS: array of resource objects allocated to process2

9: Truncate(voldist_proc1);
10: Truncate(voldist_proc2);

11: RMatrix ← LoadReport(TCR);
12: RS ← LoadResources(RM[Column(0)]);

13: P1RS ← Filter(Filter(RMatrix[Column(1)], T), RS);
14: P2RS ← Filter(Filter(RMatrix[Column(2)], T), RS);

15: if P1RS = 0 || P2RS = 0 then
16:    return −1;
17: end if

18: SIDs ← GetSEDOLs();

19: GridJobs ← [GridJob('p1', SIDs, P1RS), GridJob('p2', SIDs, P2RS)];
20: for i = 1 to 2 do
21:    Run(GridJobs[i]);
22: end for

23: return 0;
```

.3.3 Simulation Outputs

```
2010-01-24 10:16:18,112 INFO Initialising simulation.
2010-01-24 10:16:18,114 INFO Executing Grid Jobs:
report=24012010.dat threshold=0.6
2010-01-24 10:16:20,351 INFO Initialised grid_resource=1.0
fail_factors=['1:0.99', '2:0.27']
2010-01-24 10:16:20,625 INFO Initialised grid_resource=2.0
fail_factors=['1:0.99', '2:0.21']
2010-01-24 10:16:20,782 INFO Initialised grid_resource=3.0
fail_factors=['1:0.99', '2:0.26']
2010-01-24 10:16:20,901 INFO Initialised grid_resource=4.0
fail_factors=['1:0.99', '2:0.41']
2010-01-24 10:16:21,006 INFO Initialised grid_resource=5.0
fail_factors=['1:0.99', '2:0.46']
2010-01-24 10:16:21,072 INFO Initialised grid_resource=6.0
fail_factors=['1:0.99', '2:0.17']
2010-01-24 10:16:21,138 INFO Initialised grid_resource=7.0
fail_factors=['1:0.99', '2:0.14']
2010-01-24 10:16:21,203 INFO Initialised grid_resource=8.0
fail_factors=['1:0.99', '2:0.21']
2010-01-24 10:16:21,269 INFO Initialised grid_resource=9.0
fail_factors=['1:0.99', '2:0.24']
2010-01-24 10:16:21,339 INFO Initialised grid_resource=10.0
fail_factors=['1:0.99', '2:0.41']
2010-01-24 10:16:21,407 INFO Initialised grid_resource=11.0
fail_factors=['1:0.99', '2:0.01']
2010-01-24 10:16:21,480 INFO Initialised grid_resource=12.0
fail_factors=['1:0.99', '2:0.38']
2010-01-24 10:16:21,546 INFO Initialised grid_resource=13.0
fail_factors=['1:0.99', '2:0.24']
2010-01-24 10:16:21,622 INFO Initialised grid_resource=14.0
fail_factors=['1:0.99', '2:0.41']
2010-01-24 10:16:21,687 INFO Initialised grid_resource=15.0
fail_factors=['1:0.99', '2:0.32']
2010-01-24 10:16:21,755 INFO Initialised grid_resource=16.0
fail_factors=['1:0.99', '2:0.30']
2010-01-24 10:16:21,822 INFO Initialised grid_resource=17.0
fail_factors=['1:0.99', '2:0.36']
2010-01-24 10:16:21,901 INFO Initialised grid_resource=18.0
fail_factors=['1:0.99', '2:0.64']
2010-01-24 10:16:21,967 INFO Initialised grid_resource=19.0
fail_factors=['1:0.99', '2:0.36']
2010-01-24 10:16:22,033 INFO Initialised grid_resource=20.0
fail_factors=['1:0.99', '2:0.71']
```

```
2010-01-24 10:16:22,102 INFO Initialised grid_resource=21.0
fail_factors=['1:0.99', '2:0.01']
2010-01-24 10:16:22,174 INFO Initialised grid_resource=22.0
fail_factors=['1:0.99', '2:0.17']
2010-01-24 10:16:22,174 INFO Initialised 22 resources=[
(1, ['1=0.99', '2=0.27']), (2, ['1=0.99', '2=0.21']),
(3, ['1=0.99', '2=0.26']), (4, ['1=0.99', '2=0.41']),
(5, ['1=0.99', '2=0.46']), (6, ['1=0.99', '2=0.17']),
(7, ['1=0.99', '2=0.14']), (8, ['1=0.99', '2=0.21']),
(9, ['1=0.99', '2=0.24']), (10, ['1=0.99', '2=0.41']),
(11, ['1=0.99', '2=0.01']), (12, ['1=0.99', '2=0.38']),
(13, ['1=0.99', '2=0.24']), (14, ['1=0.99', '2=0.41']),
(15, ['1=0.99', '2=0.32']), (16, ['1=0.99', '2=0.30']),
(17, ['1=0.99', '2=0.36']), (18, ['1=0.99', '2=0.64']),
(19, ['1=0.99', '2=0.36']), (20, ['1=0.99', '2=0.71']),
(21, ['1=0.99', '2=0.01']), (22, ['1=0.99', '2=0.17'])]
2010-01-24 10:16:23,305 INFO total count symbols=6466
2010-01-24 10:16:23,305 INFO Initialised proc_id=p1.1.0 num_SEDOLs=293
2010-01-24 10:16:23,305 INFO Initialised proc_id=p1.2.0 num_SEDOLs=293
2010-01-24 10:16:23,305 INFO Initialised proc_id=p1.3.0 num_SEDOLs=293
2010-01-24 10:16:23,306 INFO Initialised proc_id=p1.4.0 num_SEDOLs=293
2010-01-24 10:16:23,306 INFO Initialised proc_id=p1.5.0 num_SEDOLs=293
2010-01-24 10:16:23,308 INFO Initialised proc_id=p1.6.0 num_SEDOLs=293
2010-01-24 10:16:23,308 INFO Initialised proc_id=p1.7.0 num_SEDOLs=293
2010-01-24 10:16:23,309 INFO Initialised proc_id=p1.8.0 num_SEDOLs=293
2010-01-24 10:16:23,309 INFO Initialised proc_id=p1.9.0 num_SEDOLs=293
2010-01-24 10:16:23,309 INFO Initialised proc_id=p1.10.0 num_SEDOLs=293
2010-01-24 10:16:23,309 INFO Initialised proc_id=p1.11.0 num_SEDOLs=293
2010-01-24 10:16:23,309 INFO Initialised proc_id=p1.12.0 num_SEDOLs=293
2010-01-24 10:16:23,311 INFO Initialised proc_id=p1.13.0 num_SEDOLs=293
2010-01-24 10:16:23,311 INFO Initialised proc_id=p1.14.0 num_SEDOLs=293
2010-01-24 10:16:23,312 INFO Initialised proc_id=p1.15.0 num_SEDOLs=293
2010-01-24 10:16:23,312 INFO Initialised proc_id=p1.16.0 num_SEDOLs=293
2010-01-24 10:16:23,312 INFO Initialised proc_id=p1.17.0 num_SEDOLs=293
2010-01-24 10:16:23,312 INFO Initialised proc_id=p1.18.0 num_SEDOLs=293
2010-01-24 10:16:23,313 INFO Initialised proc_id=p1.19.0 num_SEDOLs=293
2010-01-24 10:16:23,313 INFO Initialised proc_id=p1.20.0 num_SEDOLs=293
2010-01-24 10:16:23,315 INFO Initialised proc_id=p1.21.0 num_SEDOLs=293
2010-01-24 10:16:23,315 INFO Initialised proc_id=p1.22.0 num_SEDOLs=313
2010-01-24 10:16:23,315 INFO Setting schedule: [
(p1.r1=293)(p1.r2=293)(p1.r3=293)
(p1.r4=293)(p1.r5=293)(p1.r6=293)
(p1.r7=293)(p1.r8=293)(p1.r9=293)
(p1.r10=293)(p1.r11=293)(p1.r12=293)
```

```
(p1.r13=293)(p1.r14=293)(p1.r15=293)
(p1.r16=293)(p1.r17=293)(p1.r18=293)
(p1.r19=293)(p1.r20=293)(p1.r21=293)
(p1.r22=313)]
2010-01-24 10:16:23,315 INFO Initialising grid_job=p1
num_resources=22 num_SEDOLs=6466 typ_alloc_size=293
2010-01-24 10:16:23,315 INFO Setting 22 resources=
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22]
2010-01-24 10:16:23,315 INFO Initialised proc_id=p2.20.0 num_SEDOLs=6466
2010-01-24 10:16:23,315 INFO Setting schedule: [(p2.r20=6466)]
2010-01-24 10:16:23,315 INFO Initialising grid_job=p2
num_resources=1 num_SEDOLs=6466 typ_alloc_size=6466
2010-01-24 10:16:23,316 INFO Setting 1 resources=[20]
2010-01-24 10:16:23,316 INFO Executing grid_job=p1...
...
2010-01-24 10:16:27,855 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:27,898 DEBUG rid=1.0 SEDOL=0185929 fid=1 fval=0.99
2010-01-24 10:16:27,898 DEBUG do_availability
2010-01-24 10:16:27,954 DEBUG rid=1.0 SEDOL=0188371 fid=2 fval=0.27
2010-01-24 10:16:27,954 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,025 DEBUG rid=1.0 SEDOL=0190462 fid=2 fval=0.27
2010-01-24 10:16:28,025 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,082 DEBUG rid=1.0 SEDOL=0195207 fid=2 fval=0.27
2010-01-24 10:16:28,084 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,141 DEBUG rid=1.0 SEDOL=0197902 fid=2 fval=0.27
2010-01-24 10:16:28,141 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,226 DEBUG rid=1.0 SEDOL=0199049 fid=2 fval=0.27
2010-01-24 10:16:28,226 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,283 DEBUG rid=1.0 SEDOL=0201502 fid=2 fval=0.27
2010-01-24 10:16:28,283 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,342 DEBUG rid=1.0 SEDOL=0201836 fid=2 fval=0.27
2010-01-24 10:16:28,342 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,398 DEBUG rid=1.0 SEDOL=0203672 fid=2 fval=0.27
2010-01-24 10:16:28,398 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
```

```
2010-01-24 10:16:28,446 DEBUG rid=1.0 SEDOL=0207458 fid=2 fval=0.27
2010-01-24 10:16:28,446 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,512 DEBUG rid=1.0 SEDOL=0214834 fid=2 fval=0.27
2010-01-24 10:16:28,513 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,582 DEBUG rid=1.0 SEDOL=0216238 fid=2 fval=0.27
2010-01-24 10:16:28,582 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,642 DEBUG rid=1.0 SEDOL=0218461 fid=2 fval=0.27
2010-01-24 10:16:28,642 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,779 DEBUG rid=1.0 SEDOL=0227218 fid=2 fval=0.27
2010-01-24 10:16:28,780 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,845 DEBUG rid=1.0 SEDOL=0229344 fid=2 fval=0.27
2010-01-24 10:16:28,845 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:28,966 DEBUG rid=1.0 SEDOL=0230346 fid=2 fval=0.27
2010-01-24 10:16:28,966 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:29,026 DEBUG rid=1.0 SEDOL=0231888 fid=2 fval=0.27
2010-01-24 10:16:29,026 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:29,084 DEBUG rid=1.0 SEDOL=0232524 fid=2 fval=0.27
2010-01-24 10:16:29,084 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:29,125 DEBUG rid=1.0 SEDOL=0233527 fid=2 fval=0.27
2010-01-24 10:16:29,127 DEBUG do_reliability
min=1 max=73.0 rand=73.0 ratio=42.0
2010-01-24 10:16:29,184 DEBUG rid=1.0 SEDOL=0234906 fid=na fval=na
...
2010-01-24 10:25:25,634 INFO Finished grid_job=p1 execution_time=542.32
2010-01-24 10:25:25,634 INFO Executing grid_job=p2...
...
2010-01-24 10:25:42,071 DEBUG rid=20.0 SEDOL=0595368 fid=na fval=na
2010-01-24 10:25:42,134 DEBUG rid=20.0 SEDOL=0600756 fid=na fval=na
2010-01-24 10:25:42,266 DEBUG rid=20.0 SEDOL=0602729 fid=na fval=na
2010-01-24 10:25:42,326 DEBUG rid=20.0 SEDOL=0603959 fid=2 fval=0.71
2010-01-24 10:25:42,326 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:42,374 DEBUG rid=20.0 SEDOL=0604316 fid=na fval=na
2010-01-24 10:25:42,470 DEBUG rid=20.0 SEDOL=0605610 fid=na fval=na
2010-01-24 10:25:42,542 DEBUG rid=20.0 SEDOL=0609140 fid=na fval=na
```

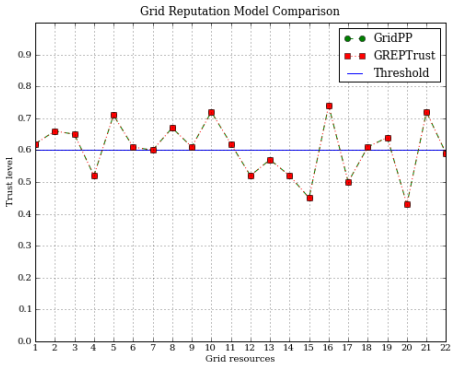
```
2010-01-24 10:25:42,584 DEBUG rid=20.0 SEDOL=0609430 fid=na fval=na
2010-01-24 10:25:42,625 DEBUG rid=20.0 SEDOL=0610841 fid=2 fval=0.71
2010-01-24 10:25:42,625 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:42,671 DEBUG rid=20.0 SEDOL=0611112 fid=2 fval=0.71
2010-01-24 10:25:42,671 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:42,726 DEBUG rid=20.0 SEDOL=0611190 fid=na fval=na
2010-01-24 10:25:42,786 DEBUG rid=20.0 SEDOL=0615200 fid=2 fval=0.71
2010-01-24 10:25:42,786 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:42,842 DEBUG rid=20.0 SEDOL=0621520 fid=2 fval=0.71
2010-01-24 10:25:42,842 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:42,987 DEBUG rid=20.0 SEDOL=0629438 fid=na fval=na
2010-01-24 10:25:43,055 DEBUG rid=20.0 SEDOL=0631303 fid=na fval=na
2010-01-24 10:25:43,142 DEBUG rid=20.0 SEDOL=0632016 fid=na fval=na
2010-01-24 10:25:43,237 DEBUG rid=20.0 SEDOL=0638939 fid=na fval=na
2010-01-24 10:25:43,325 DEBUG rid=20.0 SEDOL=0643610 fid=na fval=na
2010-01-24 10:25:43,384 DEBUG rid=20.0 SEDOL=0643900 fid=2 fval=0.71
2010-01-24 10:25:43,384 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:43,427 DEBUG rid=20.0 SEDOL=0644936 fid=na fval=na
2010-01-24 10:25:43,486 DEBUG rid=20.0 SEDOL=0646608 fid=na fval=na
2010-01-24 10:25:43,526 DEBUG rid=20.0 SEDOL=0655604 fid=na fval=na
2010-01-24 10:25:43,569 DEBUG rid=20.0 SEDOL=0661689 fid=na fval=na
2010-01-24 10:25:43,614 DEBUG rid=20.0 SEDOL=0662789 fid=na fval=na
2010-01-24 10:25:43,654 DEBUG rid=20.0 SEDOL=0664097 fid=2 fval=0.71
2010-01-24 10:25:43,655 DEBUG do_reliability
 min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:43,700 DEBUG rid=20.0 SEDOL=0664815 fid=2 fval=0.71
2010-01-24 10:25:43,700 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
2010-01-24 10:25:43,756 DEBUG rid=20.0 SEDOL=0664901 fid=na fval=na
2010-01-24 10:25:43,798 DEBUG rid=20.0 SEDOL=0665045 fid=na fval=na
2010-01-24 10:25:43,855 DEBUG rid=20.0 SEDOL=0666747 fid=na fval=na
2010-01-24 10:25:43,898 DEBUG rid=20.0 SEDOL=0667278 fid=na fval=na
2010-01-24 10:25:43,943 DEBUG rid=20.0 SEDOL=0667438 fid=na fval=na
2010-01-24 10:25:43,984 DEBUG rid=20.0 SEDOL=0668323 fid=2 fval=0.71
2010-01-24 10:25:43,984 DEBUG do_reliability
min=1 max=29.0 rand=29.0 ratio=42.0
...
2010-01-24 10:32:04,661 INFO Finished grid_job=p2 execution_time=399.03
2010-01-24 10:32:04,661 INFO Simulation completed. (946.55 secs)
```

.4 Dissection of Experimental Results

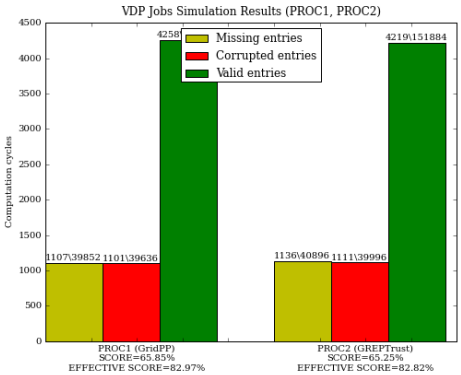
.4.1 Structure

1. B1 - Preliminary Experiments
   (a) B1S1 - Simultaneously increasing availability and decreasing reliability
      i. B1S1TC1 - ($A_\mu = 0.60, R_\mu = 0.60$)
      ii. B1S1TC2 - ($A_\mu = 0.75, R_\mu = 0.45$)
      iii. B1S1TC3 - ($A_\mu = 0.90, R_\mu = 0.30$)
2. B2 - Single factor experiments
   (a) B2S1 - Ideal availability & low reliability
      i. B2S1TC1 - ($A_\mu = 1.00, R_\mu = 0.30$)
   (b) B2S2 - Consistent availability & gradually decreasing reliability
      i. B2S2TC1 - ($A_\mu = 0.60, R_\mu = 0.60$)
      ii. B2S2TC2 - ($A_\mu = 0.60, R_\mu = 0.45$)
      iii. B2S2TC3 - ($A_\mu = 0.60, R_\mu = 0.30$)
      iv. B2S2TC4 - ($A_\mu = 0.60, R_\mu = 0.15$)
   (c) B2S3 - Consistent reliability & gradually decreasing availability
      i. B2S3TC1 - ($A_\mu = 0.60, R_\mu = 0.60$)
      ii. B2S3TC2 - ($A_\mu = 0.45, R_\mu = 0.60$)
      iii. B2S3TC3 - ($A_\mu = 0.30, R_\mu = 0.60$)
      iv. B2S3TC4 - ($A_\mu = 0.15, R_\mu = 0.60$)
3. B3 - Multi factor experiments
   (a) B3S1 - Ideal availability & low reliability
      i. B3S1TC1 - ($A_\mu = 1.00, R_\mu = 0.30$)
   (b) B3S2 - Consistent availability & gradually decreasing reliability
      i. B3S2TC1 - ($A_\mu = 0.60, R_\mu = 0.60$)
      ii. B3S2TC2 - ($A_\mu = 0.60, R_\mu = 0.45$)
      iii. B3S2TC3 - ($A_\mu = 0.60, R_\mu = 0.30$)
      iv. B3S2TC4 - ($A_\mu = 0.60, R_\mu = 0.15$)
   (c) B3S3 - Consistent reliability & gradually decreasing availability
      i. B3S3TC1 - ($A_\mu = 0.60, R_\mu = 0.60$)
      ii. B3S3TC2 - ($A_\mu = 0.45, R_\mu = 0.60$)
      iii. B3S3TC3 - ($A_\mu = 0.30, R_\mu = 0.60$)
      iv. B3S3TC4 - ($A_\mu = 0.15, R_\mu = 0.60$)

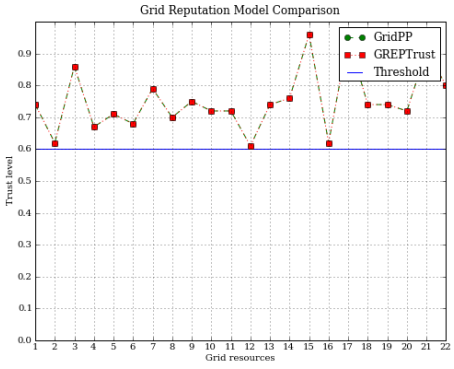
.4.2 Batch1: Preliminary Experiments



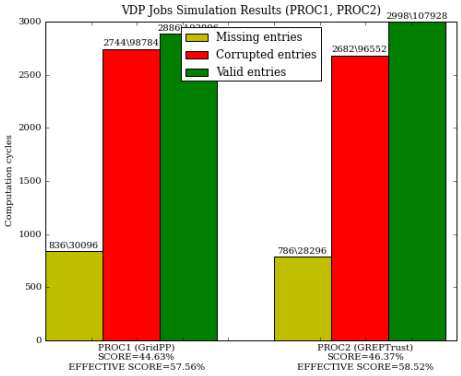
(a) Model comparison for test case: B1S1TC1



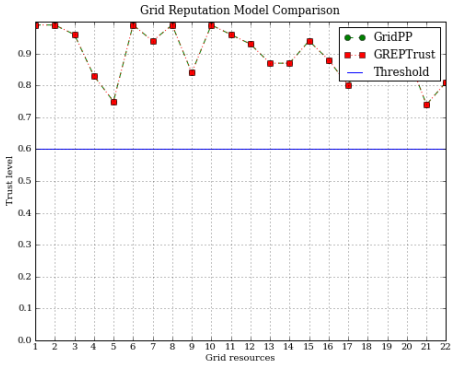
(b) Simulation results for test case: B1S1TC1



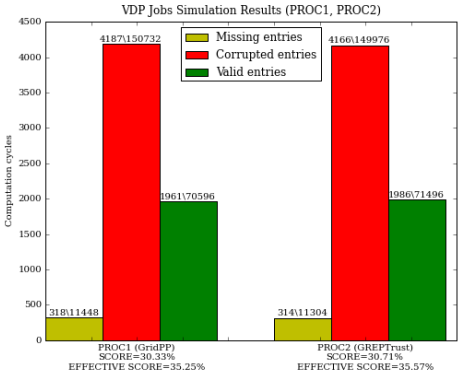
(c) Model comparison for test case: B1S1TC2



(d) Simulation results for test case: B1S1TC2



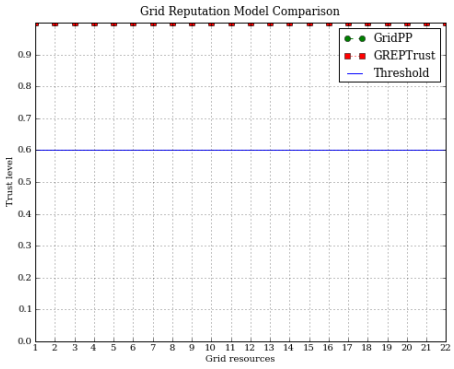
(e) Model comparison for test case: B1S1TC3



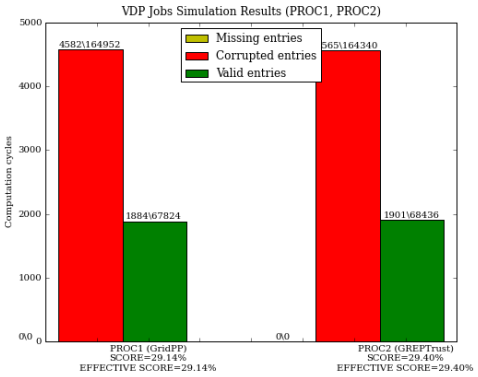
(f) Simulation results for test case: B1S1TC3

Figure 10: Execution analysis for scenario: B1S1

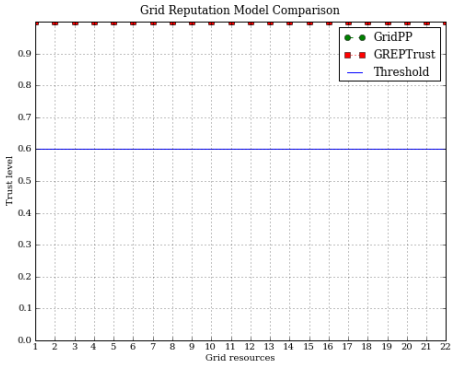
.4.3 Batch2: Single Factor Experiments



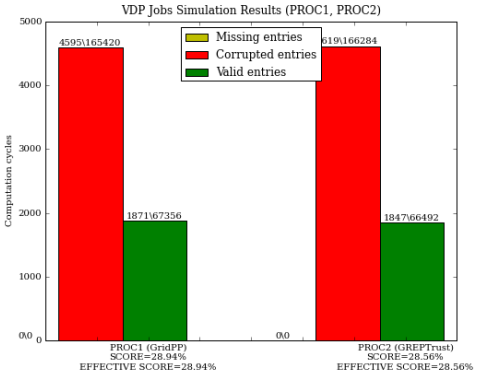
(a) Model comparison for test case: B2S1TC1-1A



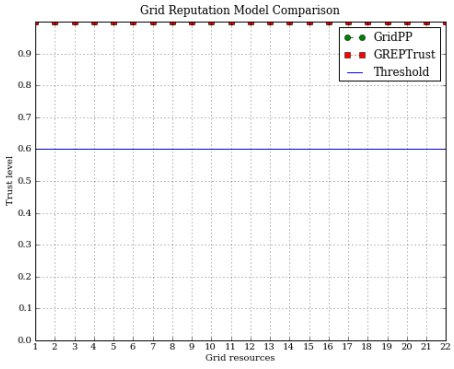
(b) Simulation results for test case: B2S1TC1-1A



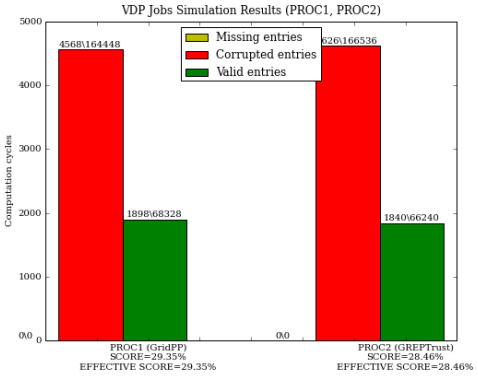
(c) Model comparison for test case: B2S1TC1-1B



(d) Simulation results for test case: B2S1TC1-1B

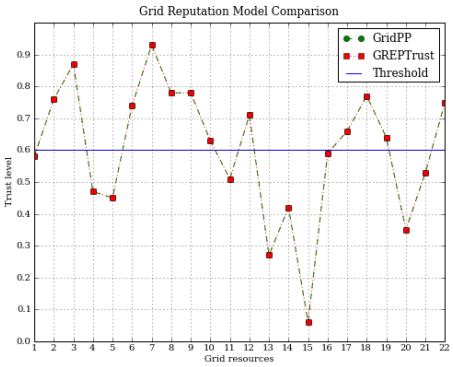


(e) Model comparison for test case: B2S1TC1-1C

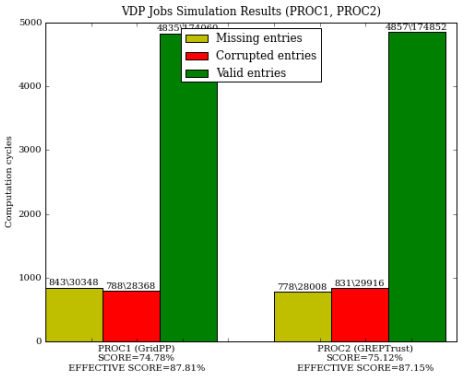


(f) Simulation results for test case: B2S1TC1-1C

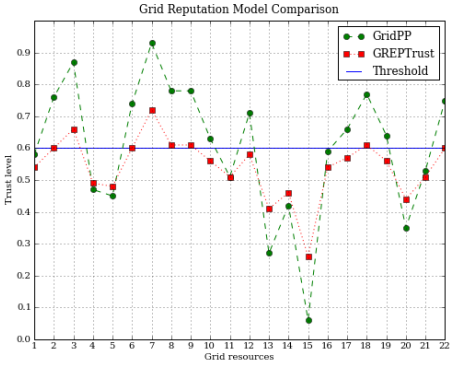
Figure 11: Execution analysis for scenario: B2S1



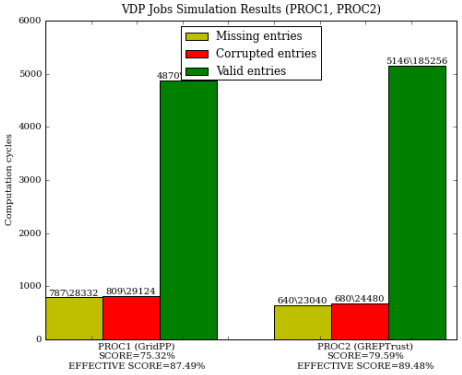
(a) Model comparison for test case: B2S2TC1-1A



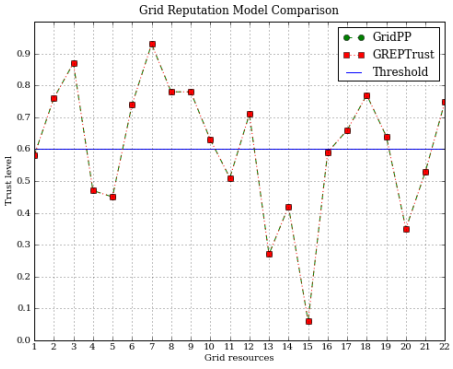
(b) Simulation results for test case: B2S2TC1-1A



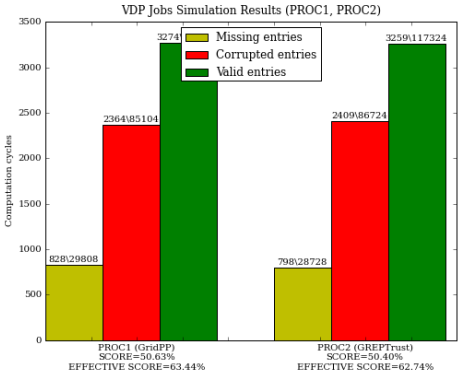
(c) Model comparison for test case: B2S2TC1-1B



(d) Simulation results for test case: B2S2TC1-1B

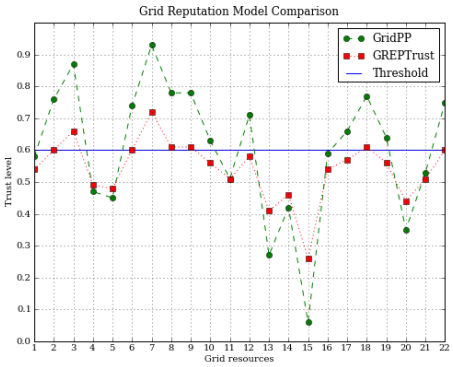


(e) Model comparison for test case: B2S2TC2-1A

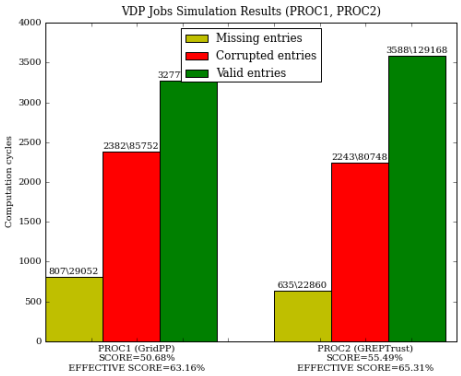


(f) Simulation results for test case: B2S2TC2-1A

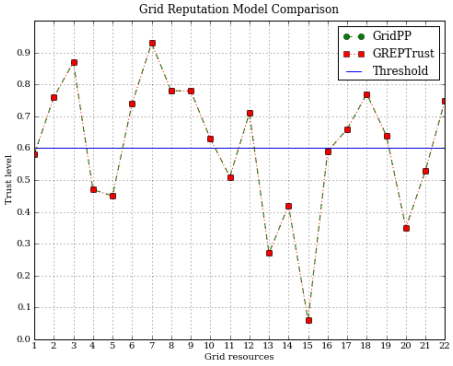
Figure 12: Execution analysis for scenario: B2S2



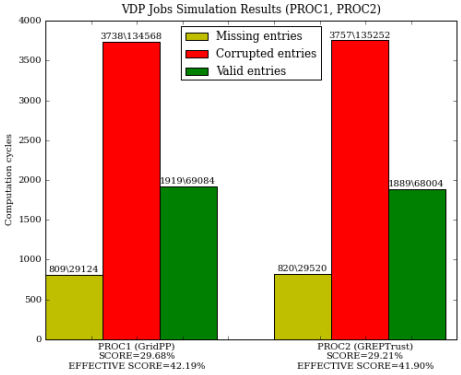
(a) Model comparison for test case: B2S2TC2-1B



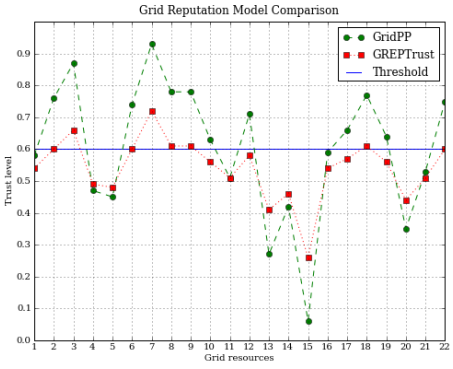
(b) Simulation results for test case: B2S2TC2-1B



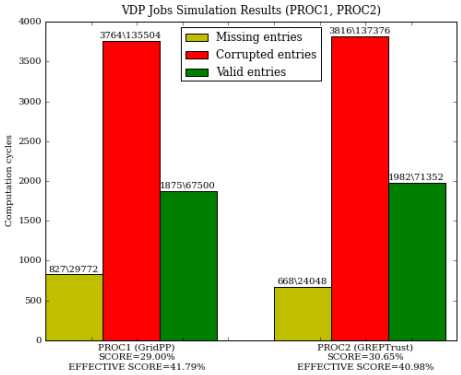
(c) Model comparison for test case: B2S2TC3-1A



(d) Simulation results for test case: B2S2TC3-1A

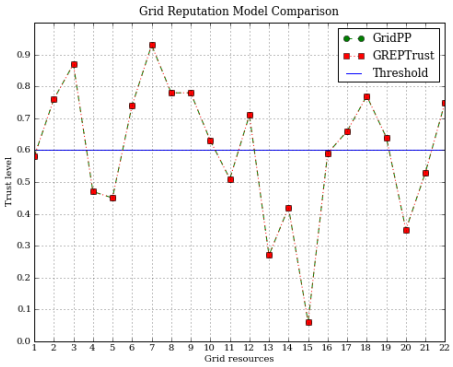


(e) Model comparison for test case: B2S2TC3-1B

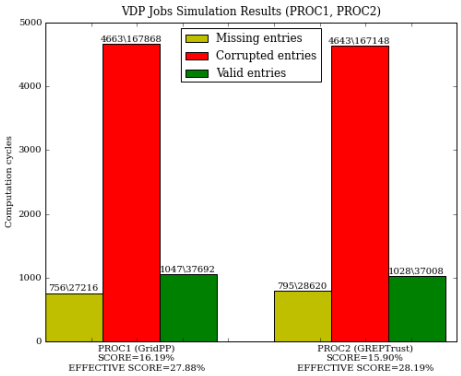


(f) Simulation results for test case: B2S2TC3-1B

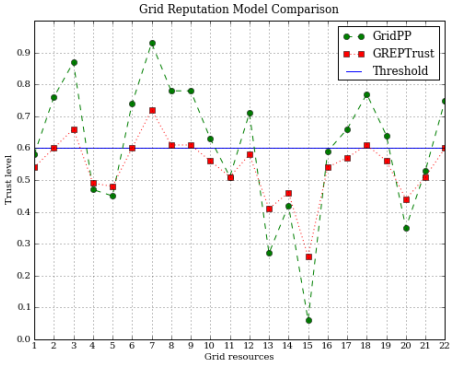
Figure 13: Execution analysis for scenario: B2S2 (continued)



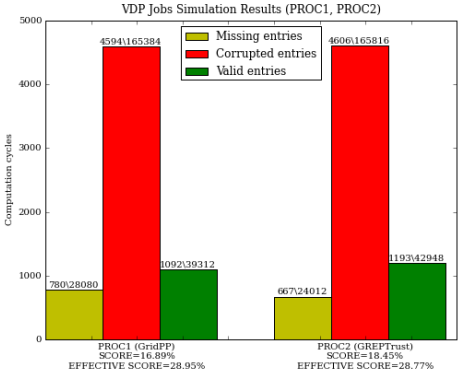
(a) Model comparison for test case: B2S2TC4-1A



(b) Simulation results for test case: B2S2TC4-1A

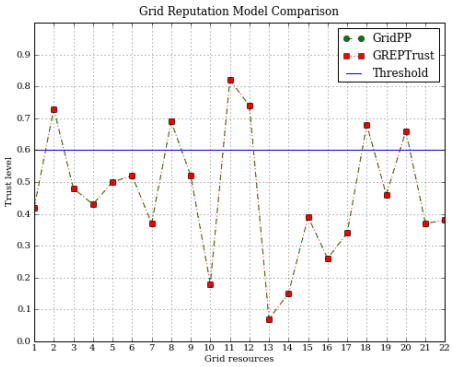


(c) Model comparison for test case: B2S2TC4-1B

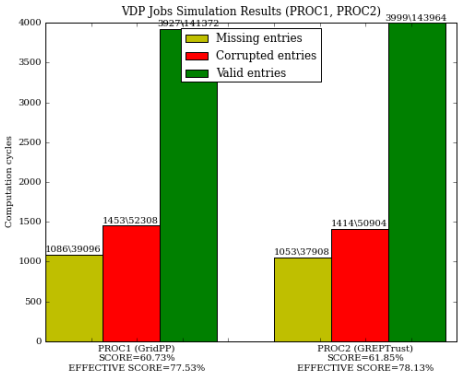


(d) Simulation results for test case: B2S2TC4-1B

Figure 14: Execution analysis for scenario: B2S2 (continued)

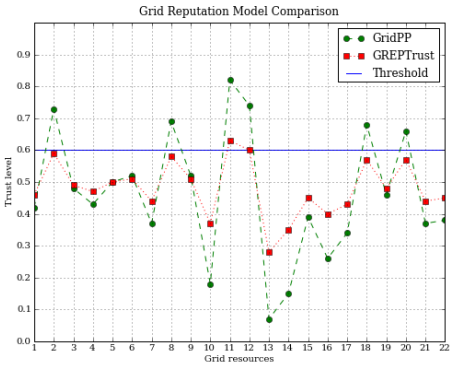


(a) Model comparison for test case: B2S3TC2-1A

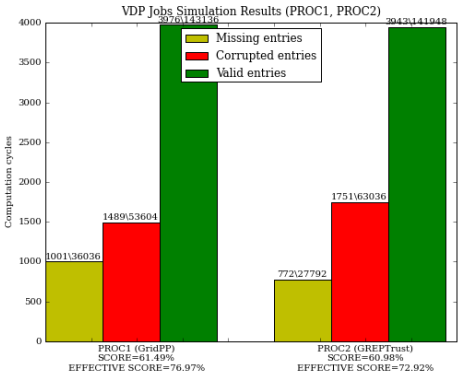


(b) Simulation results for test case: B2S3TC2-1A

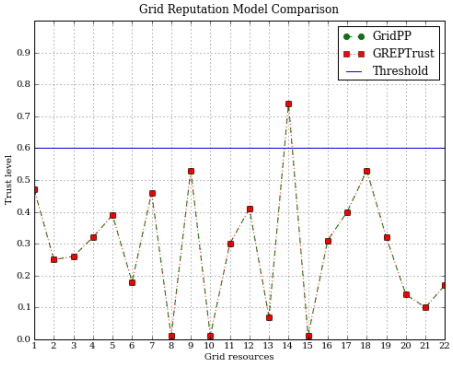
Figure 15: Execution analysis for scenario: B2S3



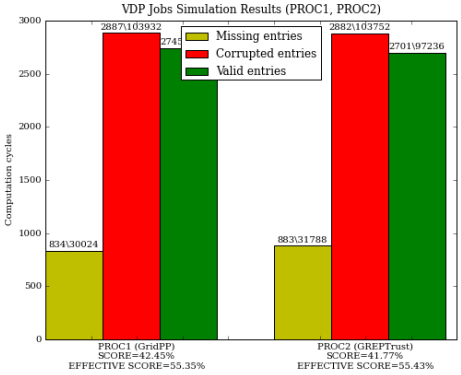
(a) Model comparison for test case: B2S3TC2-1B



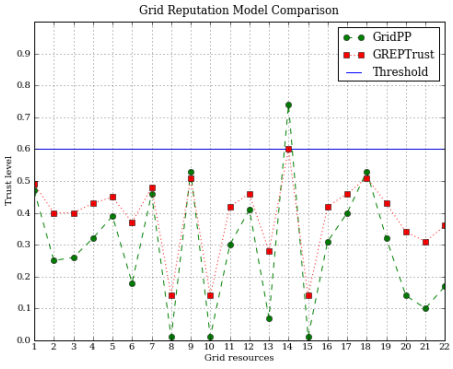
(b) Simulation results for test case: B2S3TC2-1B



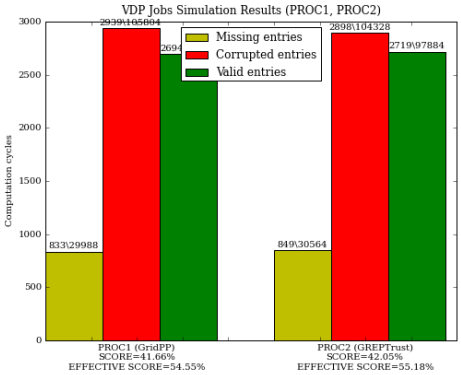
(c) Model comparison for test case: B2S3TC3-1A



(d) Simulation results for test case: B2S3TC3-1A

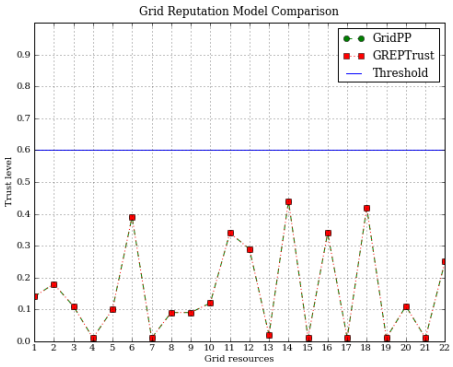


(e) Model comparison for test case: B2S3TC3-1B

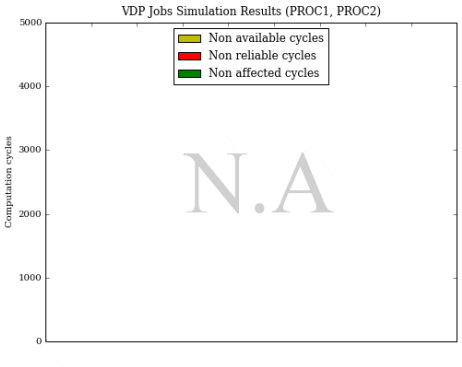


(f) Simulation results for test case: B2S3TC3-1B

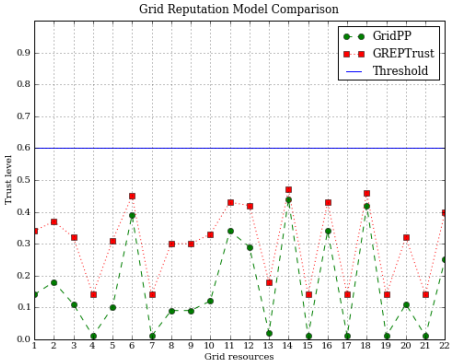
Figure 16: Execution analysis for scenario: B2S3 (continued)



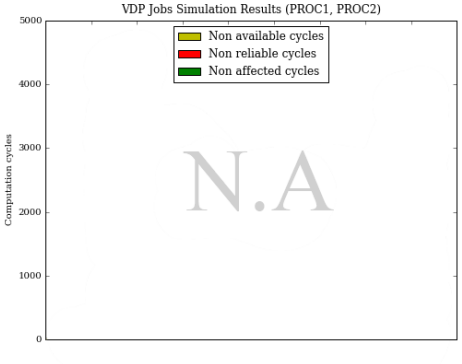
(a) Model comparison for test case: B2S3TC4-1A



(b) Simulation results for test case: B2S3TC4-1A



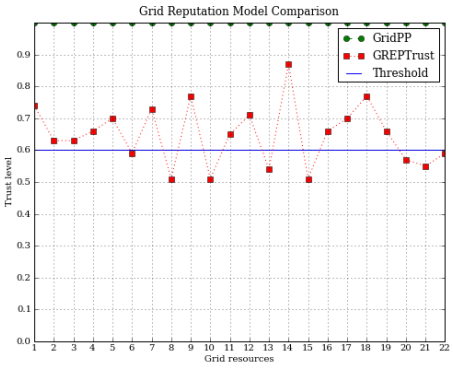
(c) Model comparison for test case: B2S3TC4-1B



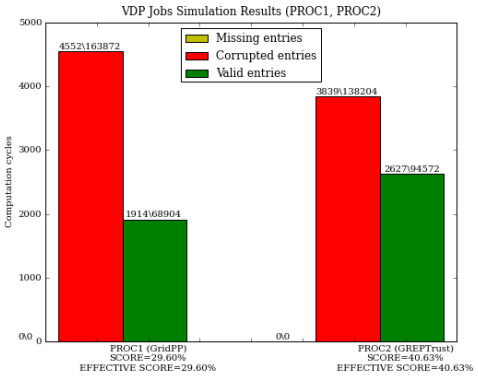
(d) Simulation results for test case: B2S3TC4-1B

Figure 17: Execution analysis for scenario: B2S3 (continued)

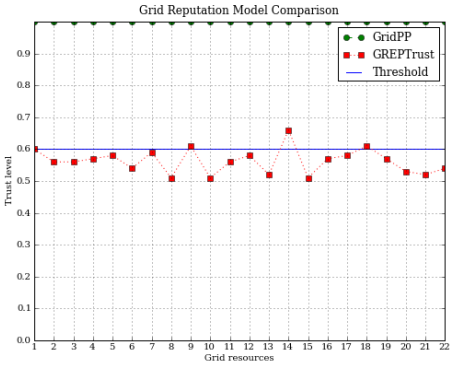
.4.4 Batch 3: Multi Factor Experiments



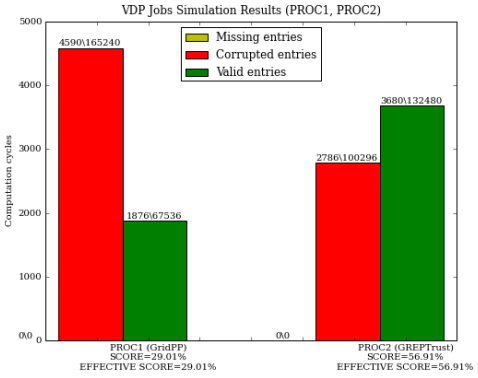
(a) Model comparison for test case execution: B3S1TC1-1A



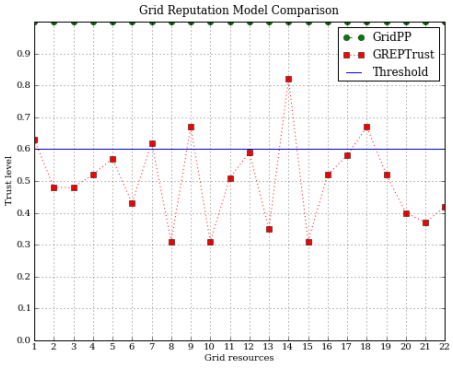
(b) Simulation results for test case execution: B1S1TC1-1A



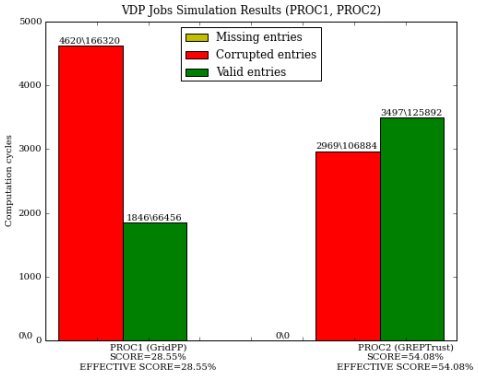
(c) Model comparison for test case execution: B3S1TC1-1B



(d) Simulation results for test case execution: B3S1TC1-1B

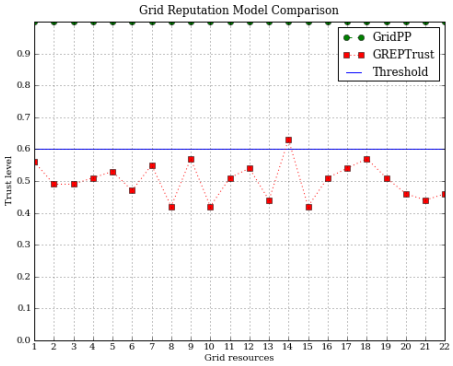


(e) Model comparison for test case execution: B3S1TC1-2A

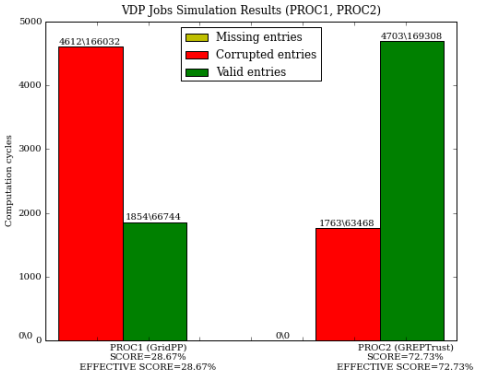


(f) Simulation results for test case execution: B3S1TC1-2A

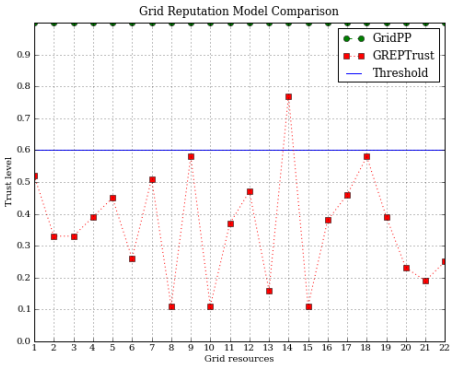
Figure 18: Execution analysis for test case: B3S1TC1



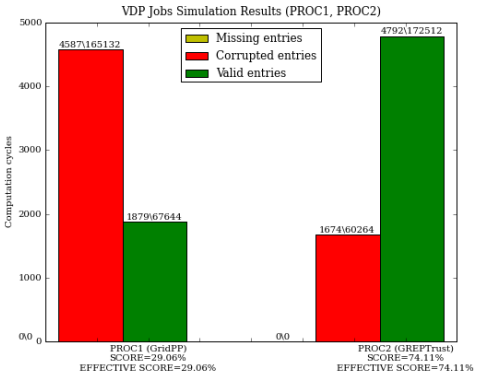
(a) Model comparison for test case execution: B3S1TC1-2B



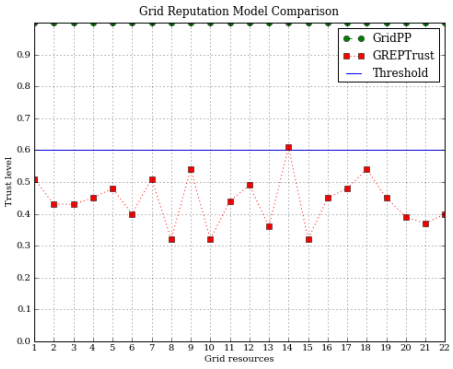
(b) Simulation results for test case execution: B3S1TC1-2B



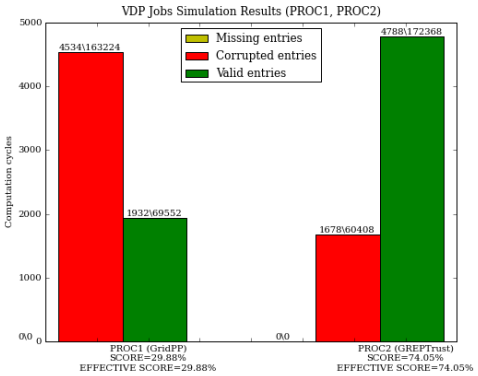
(c) Model comparison for test case execution: B3S1TC1-3A



(d) Simulation results for test case execution: B3S1TC1-3A

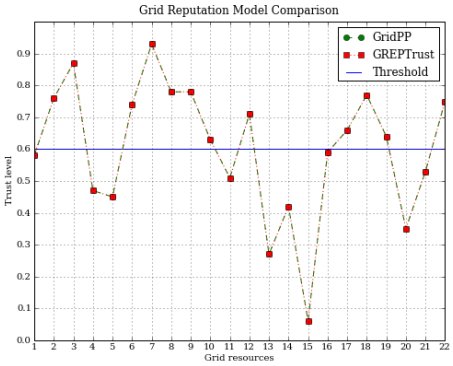


(e) Model comparison for test case execution: B3S1TC1-3B

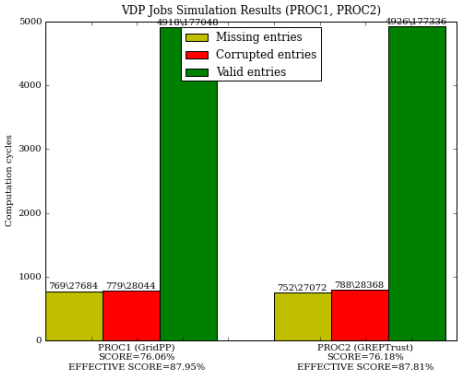


(f) Simulation results for test case execution: B3S1TC1-3B

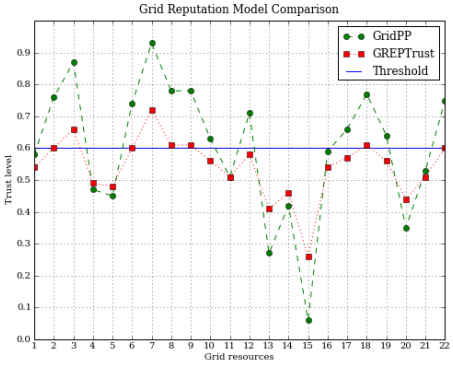
Figure 19: Execution analysis for test case: B3S1TC1 (continued)



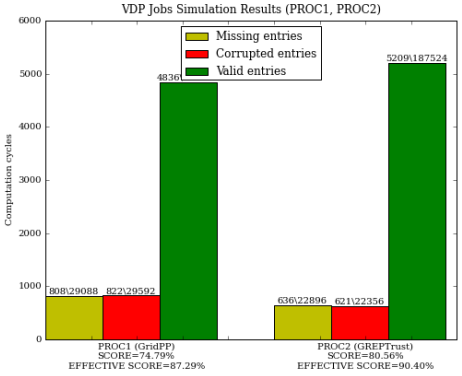
(a) Model comparison for test case execution: B3S2TC1-1A



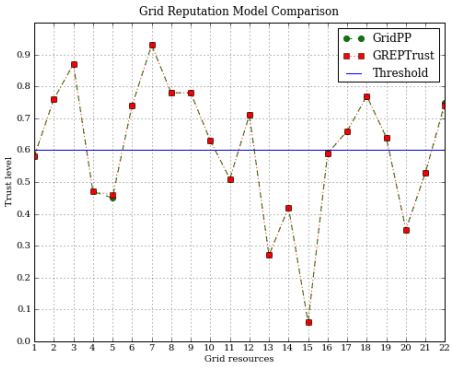
(b) Simulation results for test case execution: B3S2TC1-1A



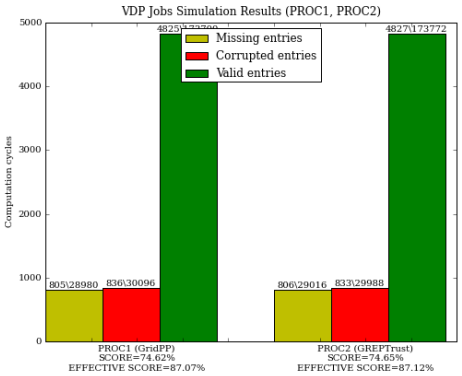
(c) Model comparison for test case execution: B3S2TC1-1B



(d) Simulation results for test case execution: B3S2TC1-1B

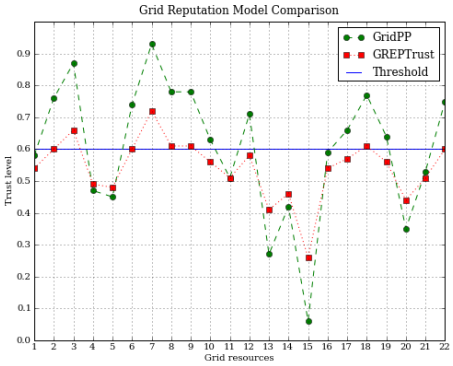


(e) Model comparison for test case execution: B3S2TC1-2A

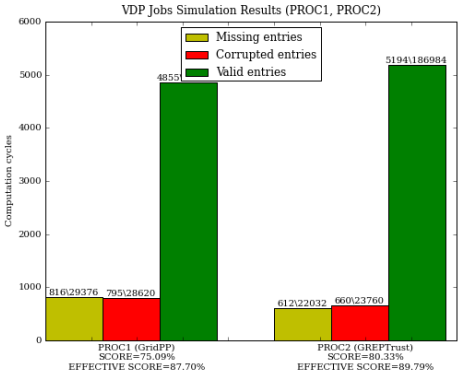


(f) Simulation results for test case execution: B3S2TC1-2A

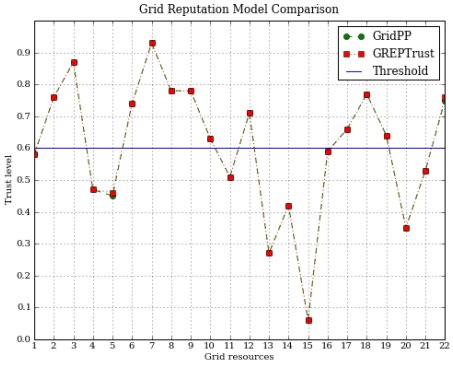
Figure 20: Execution analysis for test case: B3S2TC1



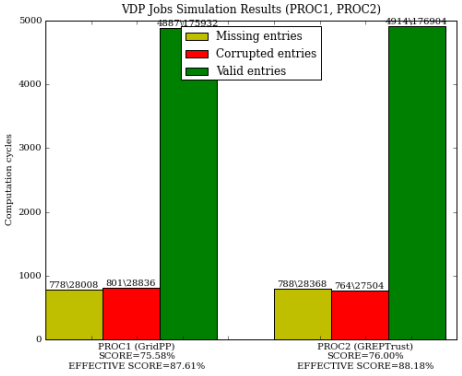
(a) Model comparison for test case execution: B3S2TC1-2B



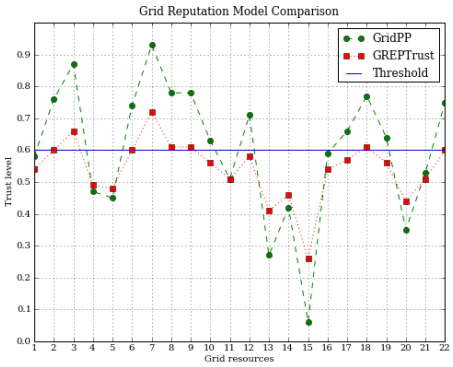
(b) Simulation results for test case execution: B3S2TC1-2B



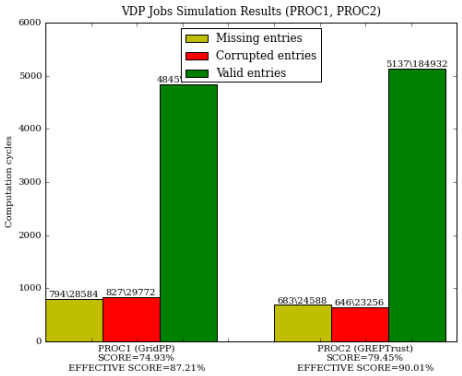
(c) Model comparison for test case execution: B3S2TC1-3A



(d) Simulation results for test case execution: B3S2TC1-3A

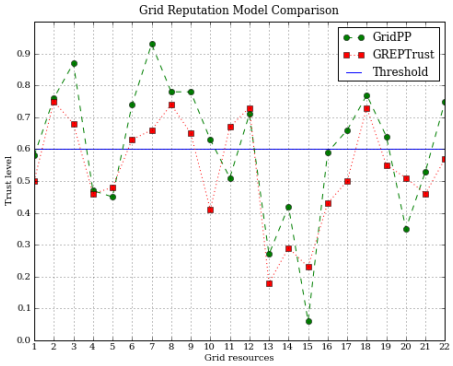


(e) Model comparison for test case execution: B3S2TC1-3B

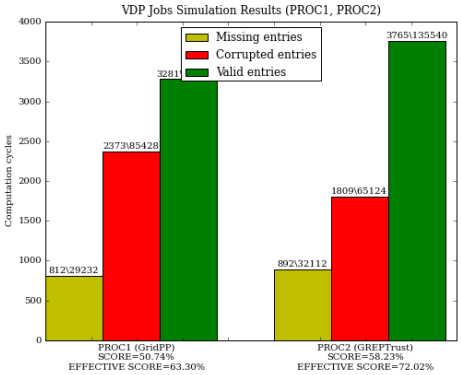


(f) Simulation results for test case execution: B3S2TC1-3B

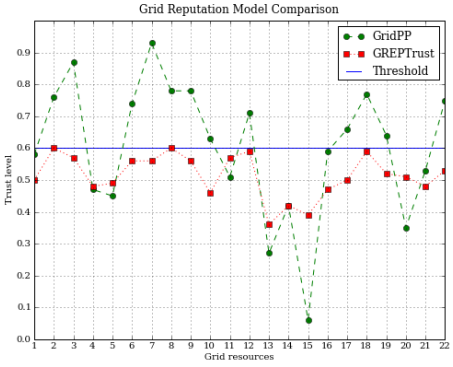
Figure 21: Execution analysis for test case: B3S2TC1 (continued)



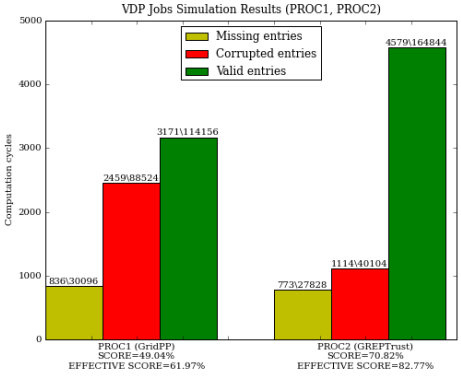
(a) Model comparison for test case execution: B3S2TC2-1A



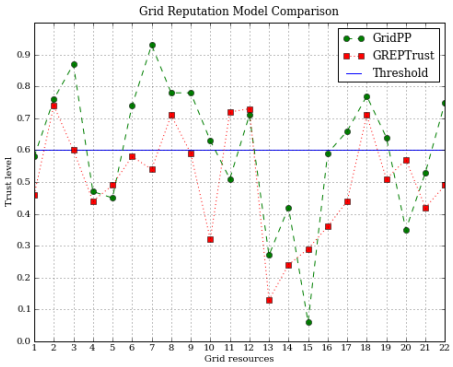
(b) Simulation results for test case execution: B2S2TC2-1A



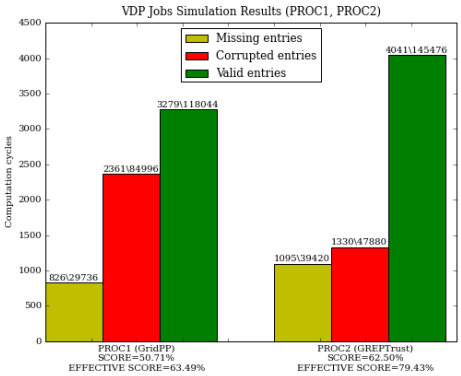
(c) Model comparison for test case execution: B3S2TC2-1B



(d) Simulation results for test case execution: B3S2TC2-1B

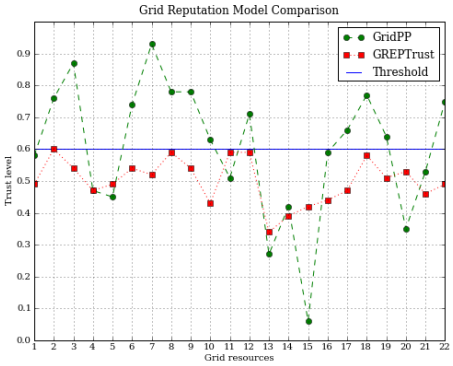


(e) Model comparison for test case execution: B3S2TC2-2A

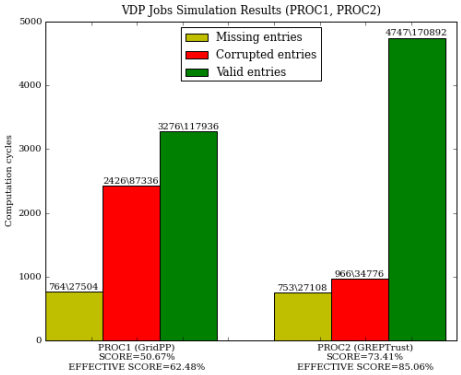


(f) Simulation results for test case execution: B3S2TC2-2A

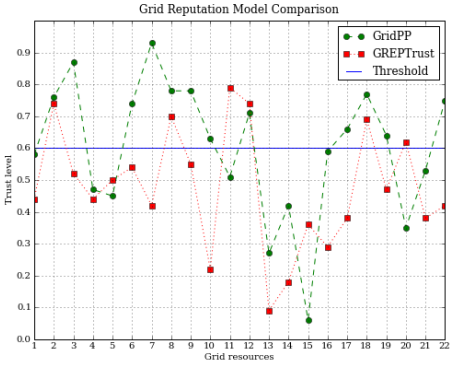
Figure 22: Execution analysis for test case: B3S2TC2



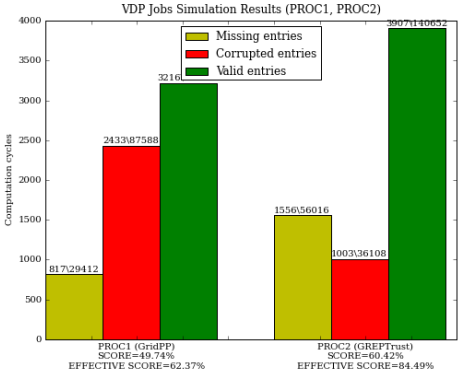
(a) Model comparison for test case execution: B3S2TC2-2B



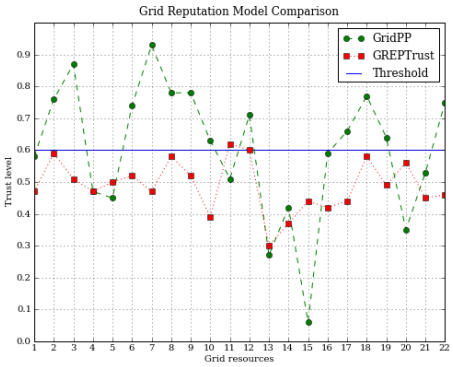
(b) Simulation results for test case execution: B3S2TC2-2B



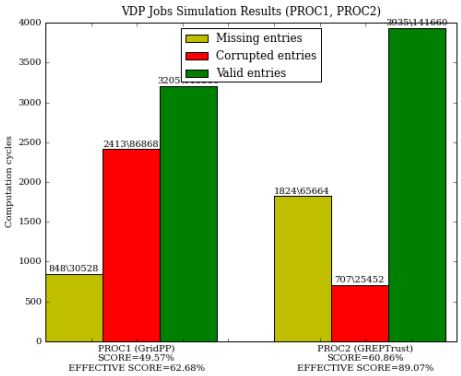
(c) Model comparison for test case execution: B3S2TC2-3A



(d) Simulation results for test case execution: B3S2TC2-3A

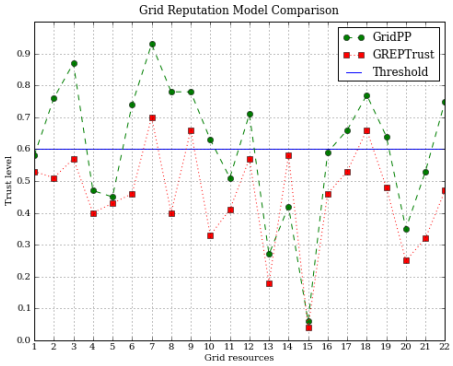


(e) Model comparison for test case execution: B3S2TC2-3B

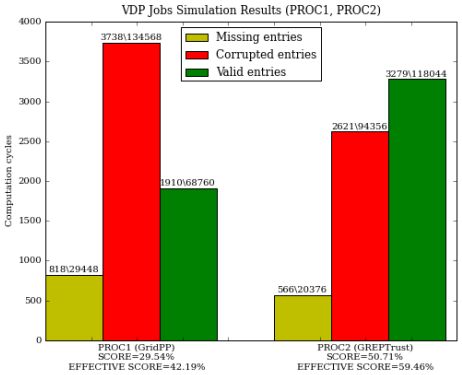


(f) Simulation results for test case execution: B3S2TC2-3B

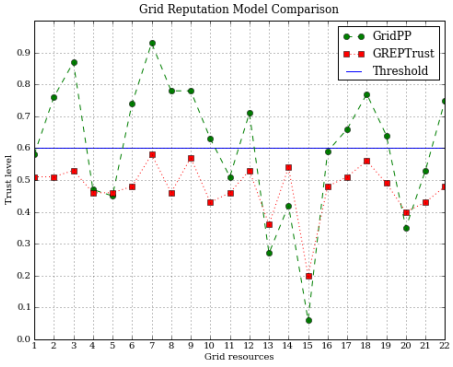
Figure 23: Execution analysis for test case: B3S2TC2 (continued)



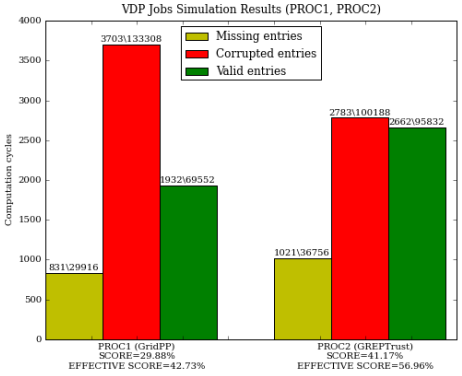
(a) Model comparison for test case execution: B3S2TC3-1A



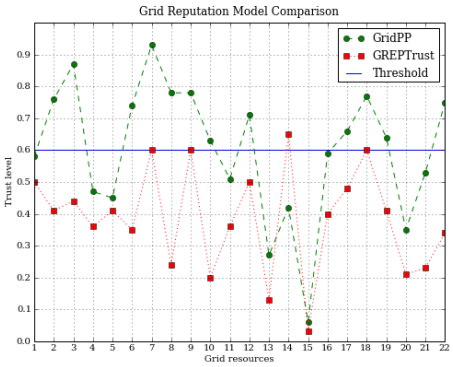
(b) Simulation results for test case execution: B3S2TC3-1A



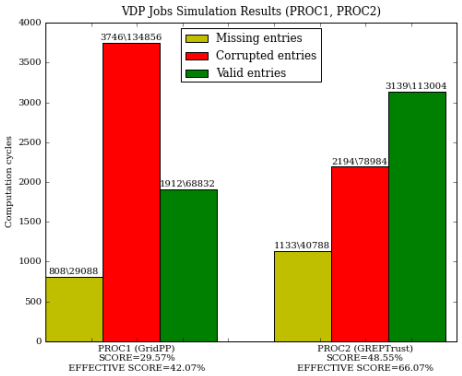
(c) Model comparison for test case execution: B3S2TC3-1B



(d) Simulation results for test case execution: B3S2TC3-1B

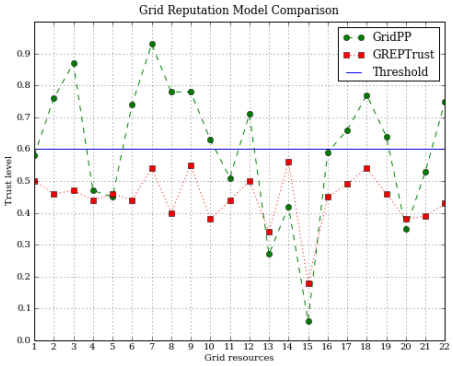


(e) Model comparison for test case execution: B3S2TC3-2A

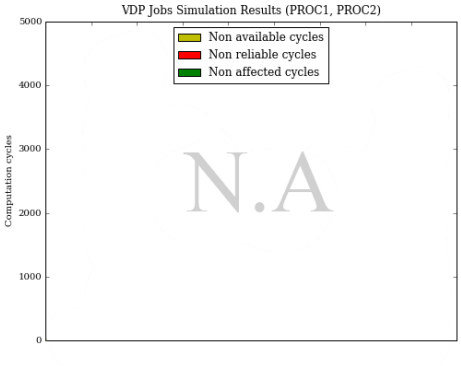


(f) Simulation results for test case execution: B3S2TC3-2A

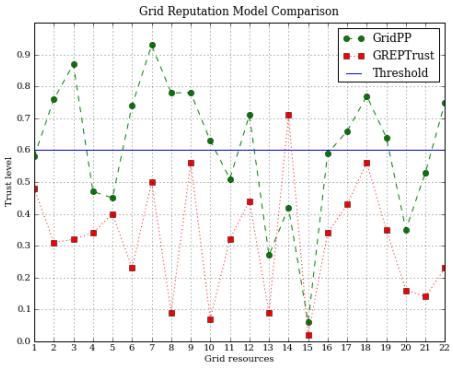
Figure 24: Execution analysis for test case: B3S2TC3



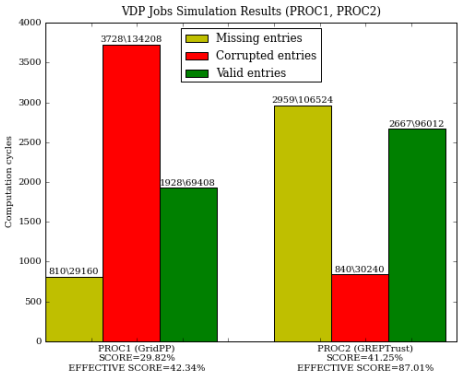
(a) Model comparison for test case execution: B3S2TC3-2B



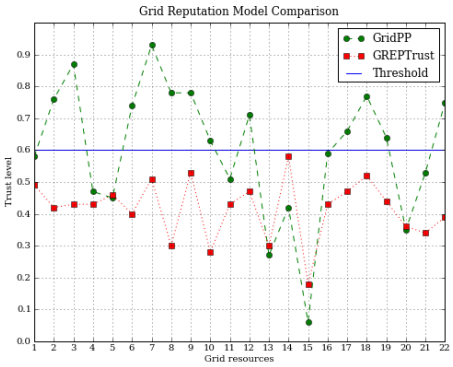
(b) Simulation results for test case execution: B3S2TC3-2B



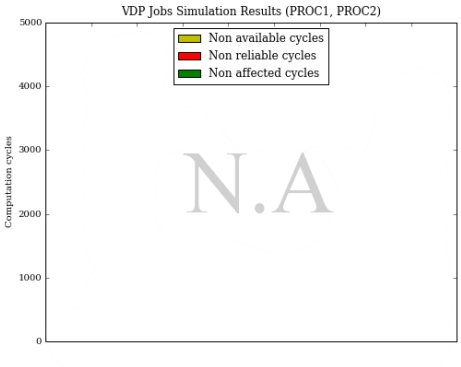
(c) Model comparison for test case execution: B3S2TC3-3A



(d) Simulation results for test case execution: B3S2TC3-3A

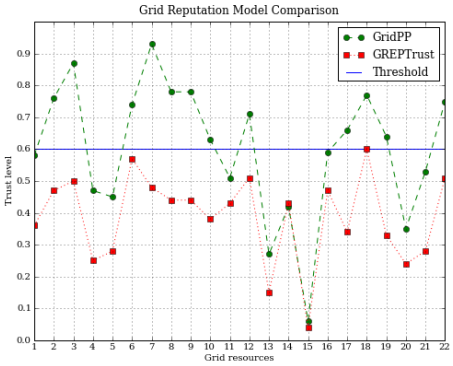


(e) Model comparison for test case execution: B3S2TC3-3B

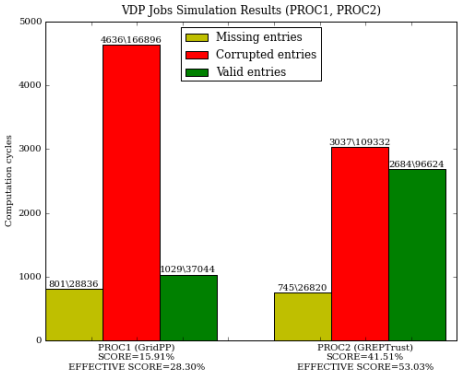


(f) Simulation results for test case execution: B3S2TC3-3B

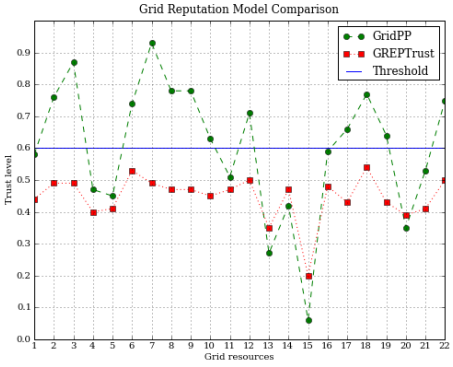
Figure 25: Execution analysis for test case: B3S2TC3 (continued)



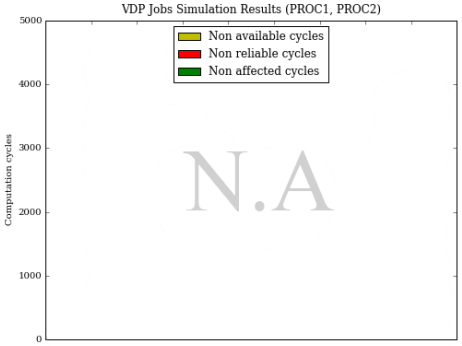
(a) Model comparison for test case execution: B3S2TC4-1A



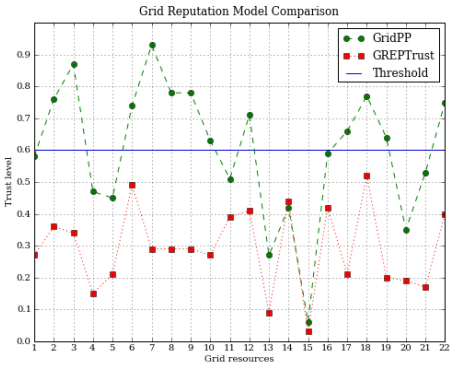
(b) Simulation results for test case execution: B1S2TC4-1A



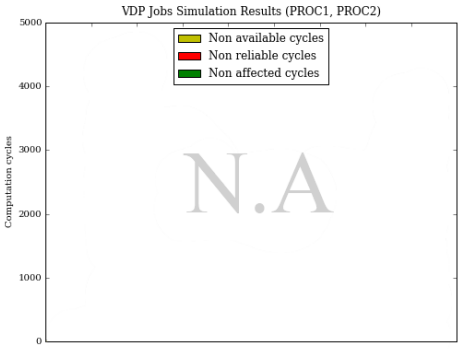
(c) Model comparison for test case execution: B3S2TC4-1B



(d) Simulation results for test case execution: B3S2TC4-1B

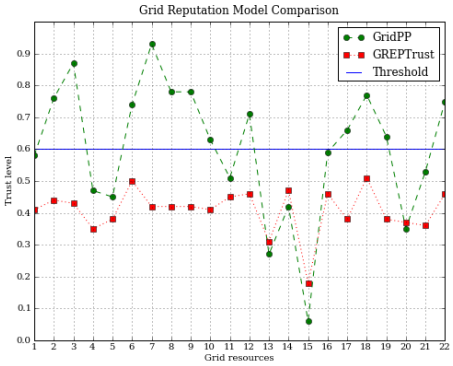


(e) Model comparison for test case execution: B3S2TC4-2A

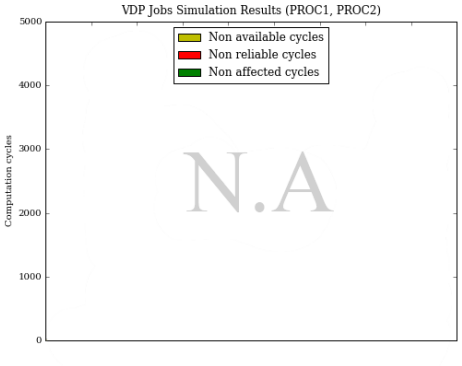


(f) Simulation results for test case execution: B3S2TC4-2A

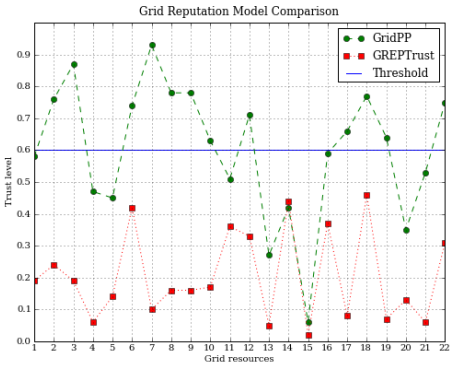
Figure 26: Execution analysis for test case: B3S2TC4



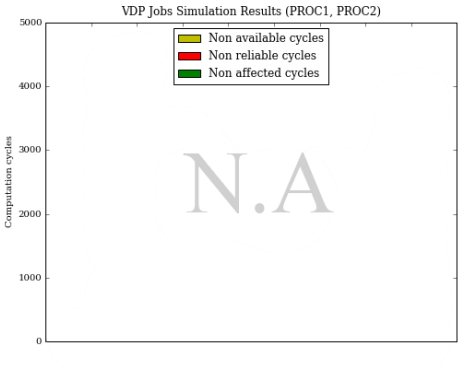
(a) Model comparison for test case execution: B3S2TC4-2B



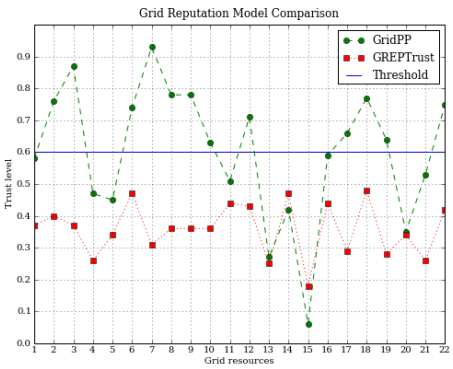
(b) Simulation results for test case execution: B3S2TC4-2B



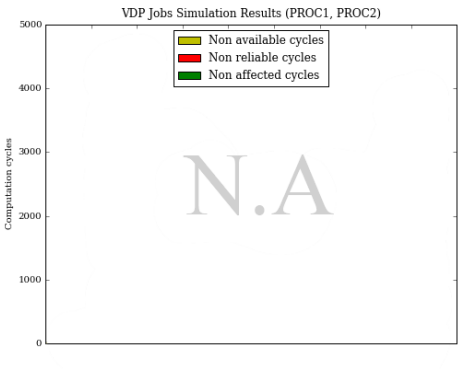
(c) Model comparison for test case execution: B3S2TC4-3A



(d) Simulation results for test case execution: B3S2TC4-3A

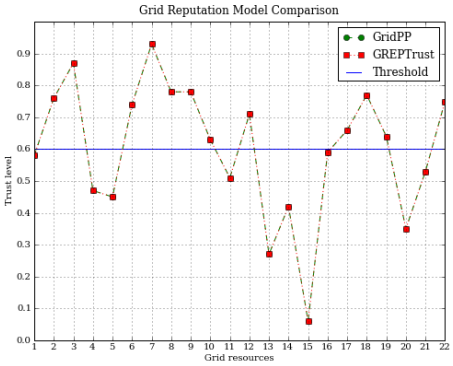


(e) Model comparison for test case execution: B3S2TC4-3B

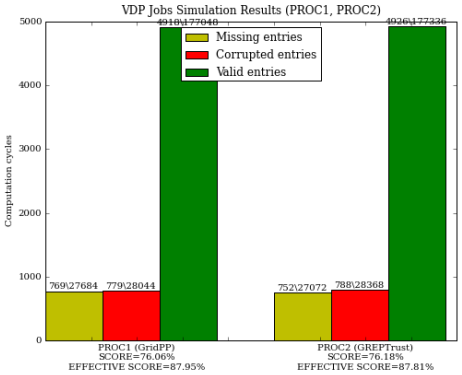


(f) Simulation results for test case execution: B3S2TC4-3B

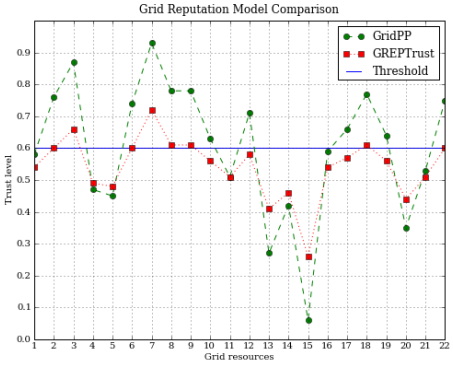
Figure 27: Execution analysis for test case: B3S2TC4 (continued)



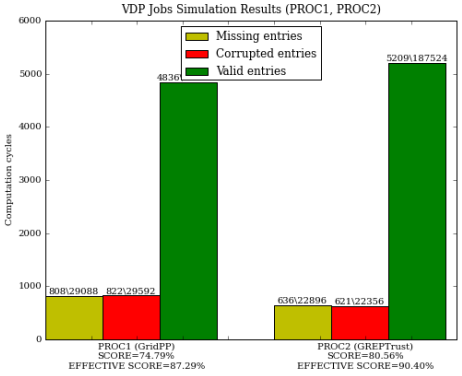
(a) Model comparison for test case execution: B3S3TC1-1A



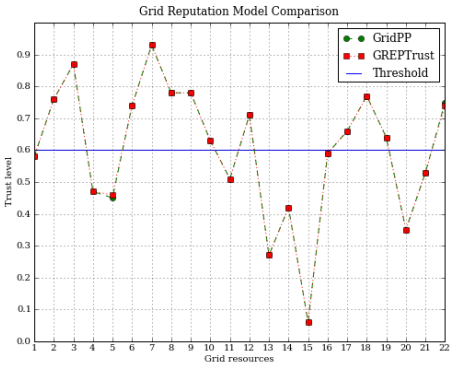
(b) Simulation results for test case execution: B3S3TC1-1A



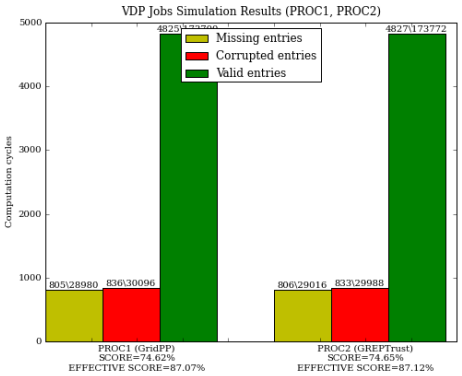
(c) Model comparison for test case execution: B3S3TC1-1B



(d) Simulation results for test case execution: B3S3TC1-1B

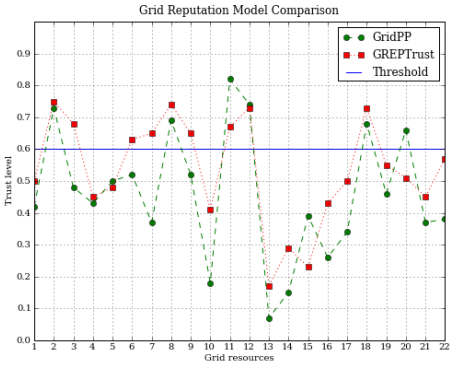


(e) Model comparison for test case execution: B3S3TC1-2A

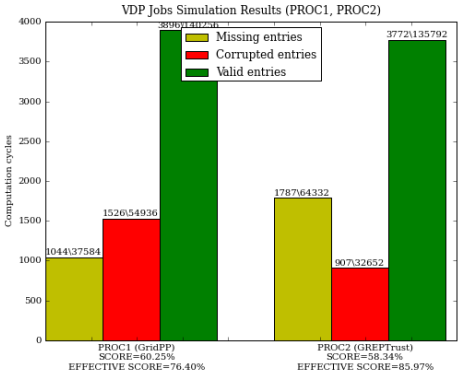


(f) Simulation results for test case execution: B3S3TC1-2A

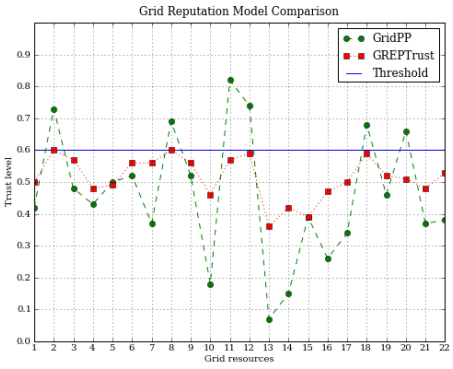
Figure 28: Execution analysis for test case: B3S3TC1



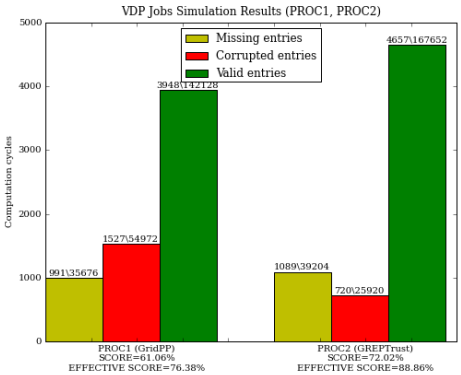
(a) Model comparison for test case execution: B3S3TC2-1A



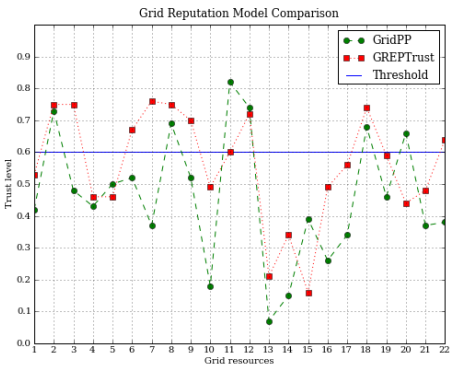
(b) Simulation results for test case execution: B3S3TC2-1A



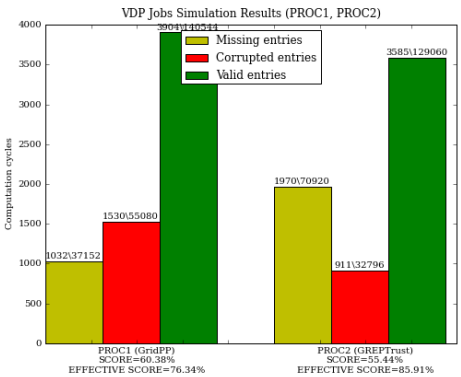
(c) Model comparison for test case execution: B3S3TC2-1B



(d) Simulation results for test case execution: B3S3TC2-1B

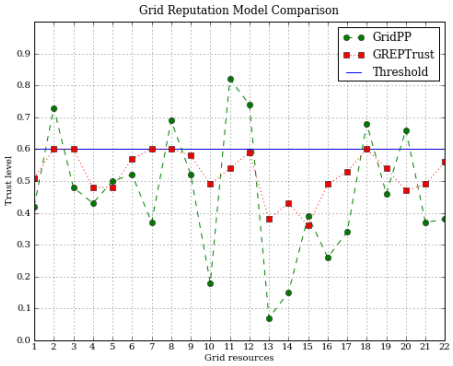


(e) Model comparison for test case execution: B3S3TC2-2A

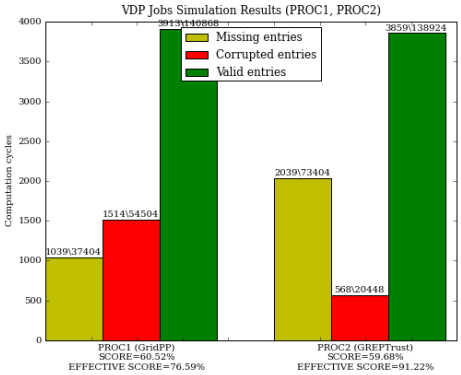


(f) Simulation results for test case execution: B3S3TC2-2A

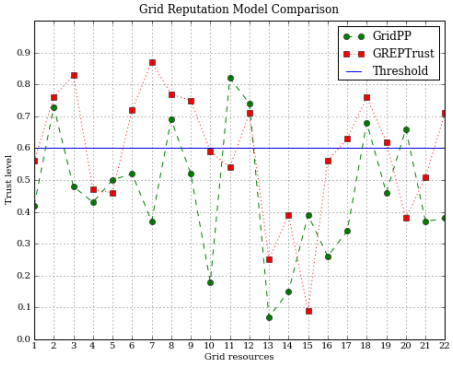
Figure 29: Execution analysis for test case: B3S3TC2



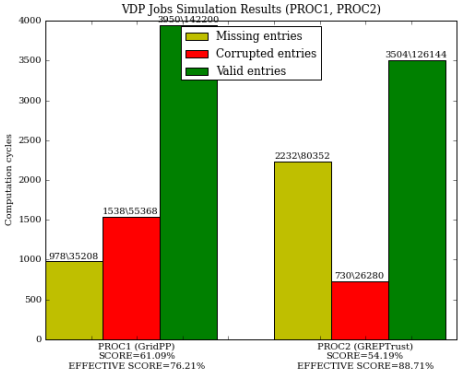
(a) Model comparison for test case execution: B3S3TC2-2B



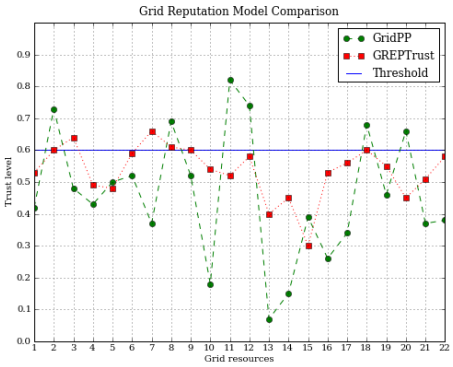
(b) Simulation results for test case execution: B3S3TC2-2B



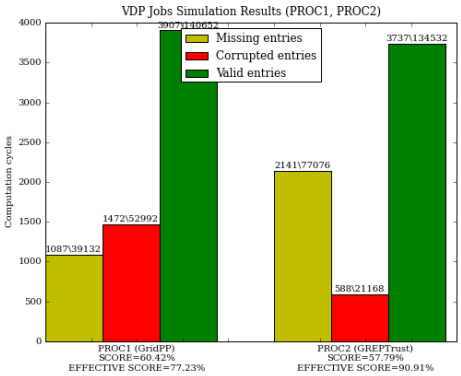
(c) Model comparison for test case execution: B3S3TC2-3A



(d) Simulation results for test case execution: B3S3TC2-3A

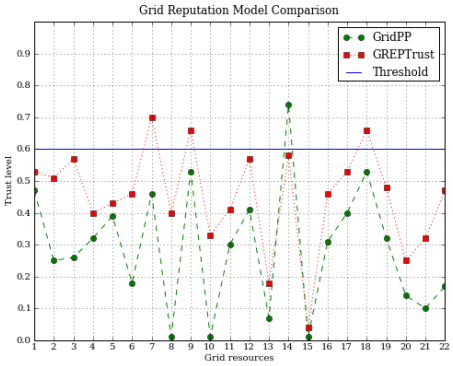


(e) Model comparison for test case execution: B3S3TC2-3B

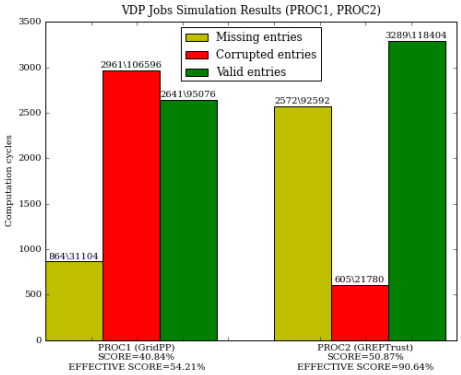


(f) Simulation results for test case execution: B3S3TC2-3B

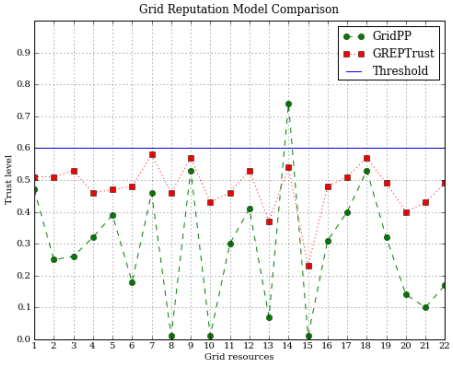
Figure 30: Execution analysis for test case: B3S3TC2 (continued)



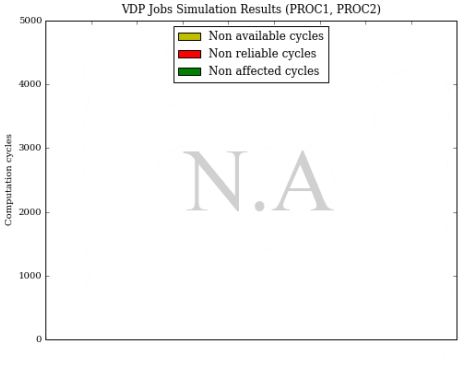
(a) Model comparison for test case execution: B3S3TC3-1A



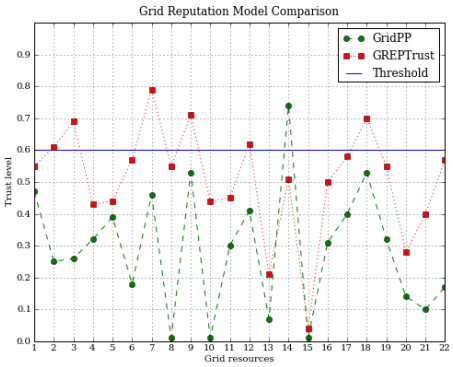
(b) Simulation results for test case execution: B3S3TC3-1A



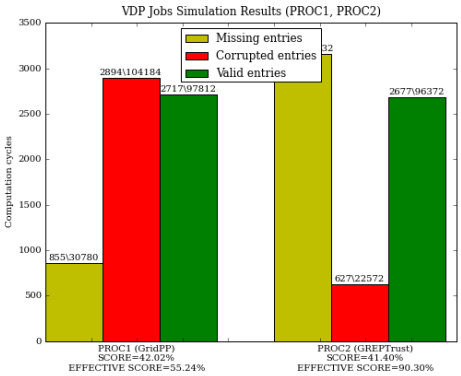
(c) Model comparison for test case execution: B3S3TC3-1B



(d) Simulation results for test case execution: B3S3TC3-1B

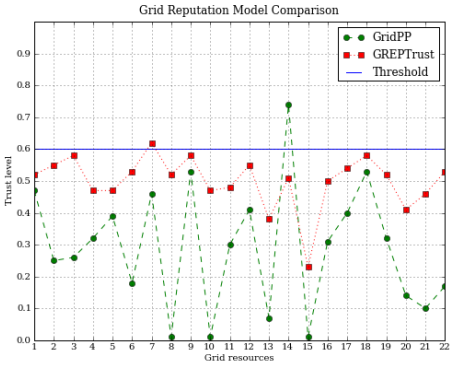


(e) Model comparison for test case execution: B3S3TC3-2A

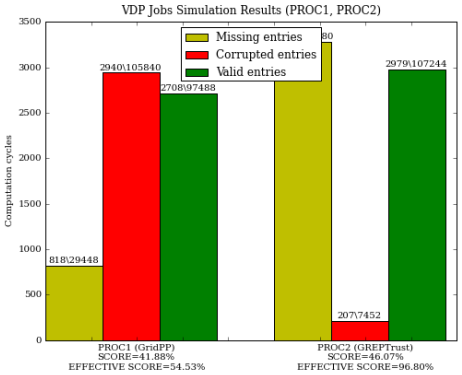


(f) Simulation results for test case execution: B3S3TC3-2A

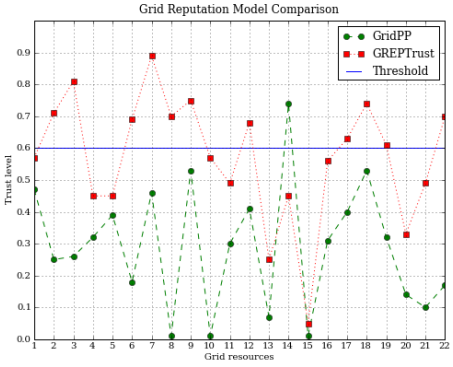
Figure 31: Execution analysis for test case: B3S3TC3



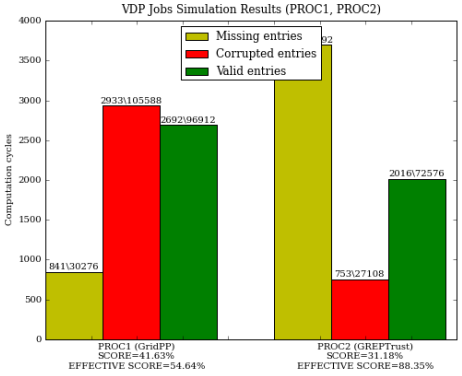
(a) Model comparison for test case execution: B3S3TC3-2B



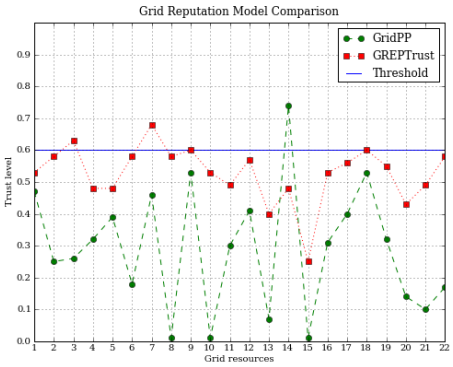
(b) Simulation results for test case execution: B3S3TC3-2B



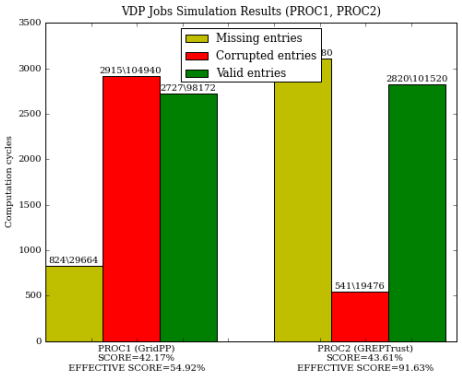
(c) Model comparison for test case execution: B3S3TC3-3A



(d) Simulation results for test case execution: B3S3TC3-3A

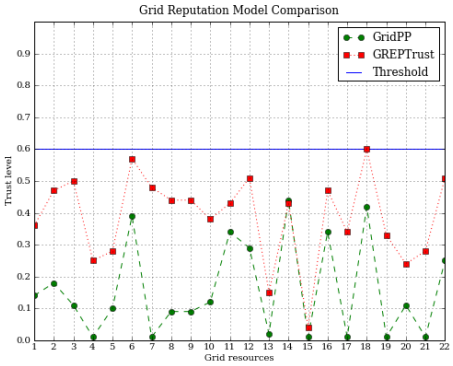


(e) Model comparison for test case execution: B3S3TC3-3B

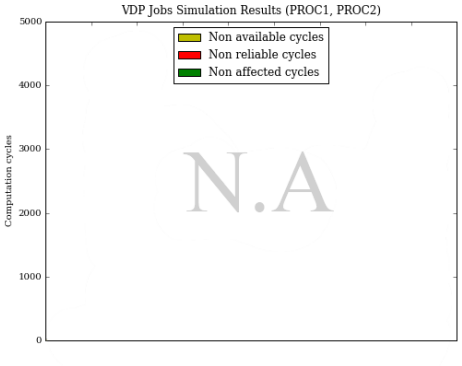


(f) Simulation results for test case execution: B3S3TC3-3B

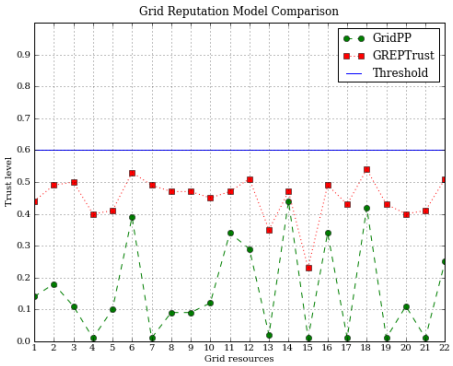
Figure 32: Execution analysis for test case: B3S3TC3 (continued)



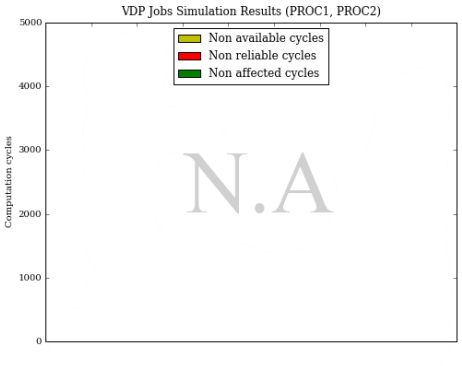
(a) Model comparison for test case execution: B3S3TC4-1A



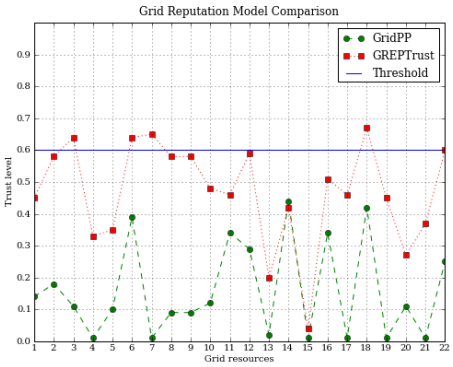
(b) Simulation results for test case execution: B3S3TC4-1A



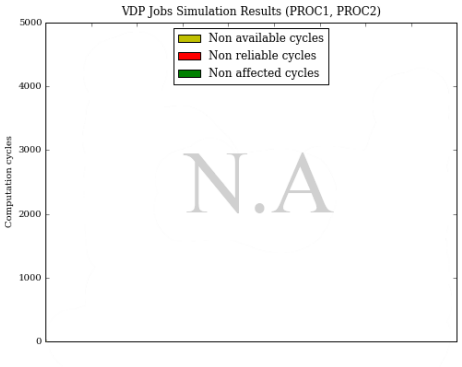
(c) Model comparison for test case execution: B3S3TC4-1B



(d) Simulation results for test case execution: B3S3TC4-1B

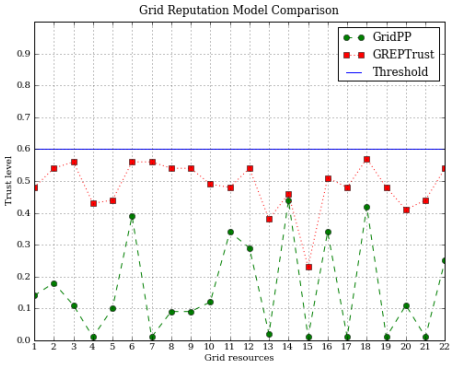


(e) Model comparison for test case execution: B3S3TC4-2A

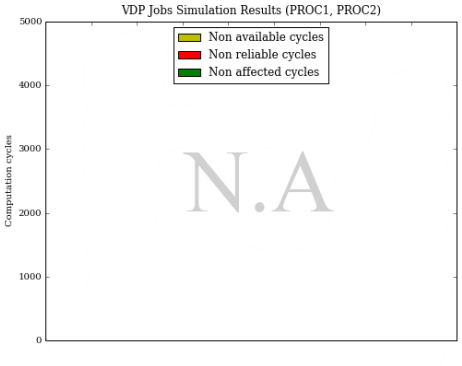


(f) Simulation results for test case execution: B3S3TC4-2A

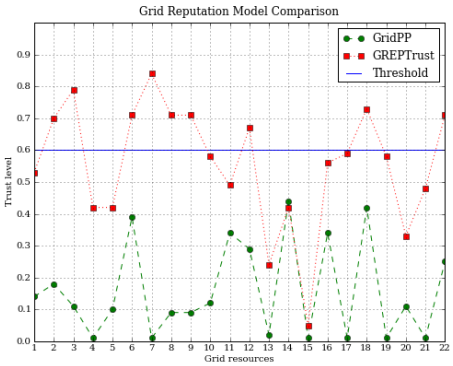
Figure 33: Execution analysis for test case: B3S3TC4



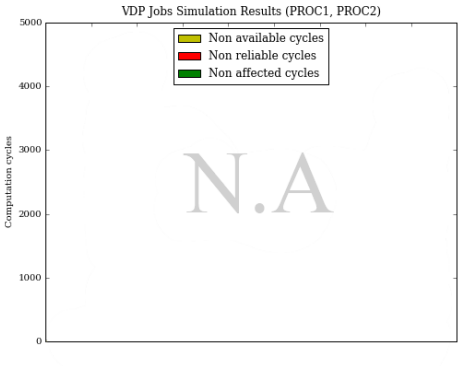
(a) Model comparison for test case execution: B3S3TC4-2B



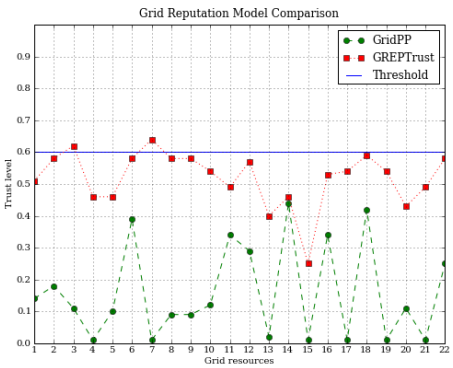
(b) Simulation results for test case execution: B3S3TC4-2B



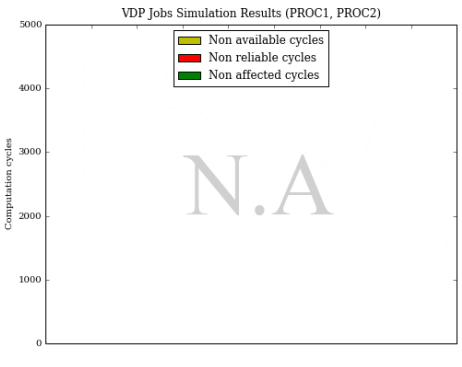
(c) Model comparison for test case execution: B3S3TC4-3A



(d) Simulation results for test case execution: B3S3TC4-3A



(e) Model comparison for test case execution: B3S3TC4-3B



(f) Simulation results for test case execution: B3S3TC4-3B

Figure 34: Execution analysis for test case: B3S3TC4 (continued)