Seniority as a Metric in Reputation Systems for E-Commerce

Catherine Cormier

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MSc in E-Business Technology

University of Ottawa

Abstract

In order to succeed, it is imperative that all e-commerce systems include an effective and reliable trust and reputation modeling system. This is particularly true of decentralized e-commerce systems in which autonomous software engage in commercial transactions. Many researchers have sought to overcome the complexities of modeling a subjective, human concept like trust, resulting in several trust and reputation models.

While these models each present a unique offering and solution to the problem, several issues persist. Most of the models require direct experience in the e-commerce system in order to make effective trust decisions. This leaves new agents and agents who only casually use the e-commerce system vulnerable. Additionally, the reputation ratings of agents who are relatively new to the system are often indistinguishable from scores for poorly performing agents. Finally, more tactics to defend against agents who exploit the characteristics of the open, distributed system for their own malicious needs are required.

To address these issues, a new metric is devised and presented: *seniority*. Based on agent age and activity level within the e-commerce system, seniority provides a means of judging the credibility of other agents with little or no prior experience in the system. As the results of experimental analysis reveals, employing a reputation model that uses seniority provides considerable value to agents who are new agents, casual buyer agents and all other purchasing agents in the e-commerce system. This new metric therefore offers a significant contribution toward the development of enhanced and new trust and reputation models for deployment in real-world distributed e-commerce environments.

Acknowledgements

I would like to extend my sincere gratitude to my thesis supervisor, Dr. Thomas Tran, for his time and direction over the course of this work. His kindness, patience and understanding have been invaluable to me and received with deepest thanks as I have navigated the challenge of completing this thesis.

I would also like to thank and recognize my parents for their unwavering love and support; without their unending help this project would not be complete. I am also immensely grateful to my brother for his continuous friendship and mentorship which have helped to see me through so much, including this work.

Finally, I'd like to express my overwhelming appreciation to my husband Shawn and my daughters, Claire and Elissa—both of whom were born over the course of the work on this thesis—for the personal sacrifices that they have made so that I may achieve my goals. They are truly my inspiration.

Table of Contents

List of Figures

List of Tables

List of Notations and Abbreviations

A	Set of advisor agents
$A_{trusted}$	Set of advisor agents that a buyer agent trusts
$A_{untrusted}$	Set of advisor agents that a buyer agent does not trust
$A_{unrated}$	Set of advisor agents for which a buyer agent does not have trust information
a_i	*i*th advisor agent in the advisor set *A*
ADV	Set of all advisor agents in marketplace *M*
adv_i	*i*th advisor agent in the set *ADV*
AR_a^v	Advisor Rating provided by advisor agent *a* regarding vendor agent *v*
B	Set of buyer agents
BL_b	Set of advisor agents blacklisted by buyer agent *b*
b_i	*i*th buyer agent in the buyer set *B*
C	Set of candidate advisor agents from which a buyer agent constructs its advisor list
CR_b^a	Credibility rating attributed to advisor *a* by buyer agent *b*
CS_b^a	Total credibility score attributed to advisor *a* by buyer agent *b*
$\gamma_{v_{selected}}$	Advice table with records of advice given about vendor $v_{selected}$
$id(a)$	Unique identification number for agent *a*
IT_b^v	Internal trust of buying agent *b* in vendor agent *v*
$l_a(t)$	Agent lifetime of agent *a* at time *t*
λ_{b_i}	Advisor rating table maintained by buyer agent b_i with records of advisor credibility ratings
M	Electronic marketplace in the seniority trust model
MAS	Multi-Agent System

$n_a(t)$	Total number of transactions in which agent *a* has participated by time *t*
ω	Credibility threshold
$R_b^v(t)$	Total reputation perceived by buyer *b* regarding vendor *v* at time *t*
$s_a(t)$	Seniority of agent *a* at time *t*
SR_b^v	Satisfaction Rating attributed to *v* by *b* based on their most recent transaction
Θ	Trust threshold
τ_a	Timestamp for time at which agent *a* entered MAS
$T_b^v(t)$	Total trust of buying agent *b* in vendor agent *v* at time *t*
t	Timestamp for current time
V	Set of vendor agents
$V_{trusted}$	Set of vendor agents that a buyer agent trusts
$V_{untrusted}$	Set of vendor agents that a buyer agent does not trust
$V_{unrated}$	Set of vendor agents for which a buyer agent does not have trust information
v_i	*i*th vendor agent in the vendor set *V*
$v_{selected}$	Vendor selected by a buyer agent as a supplier for a transaction
W_b^v	Set of advisor agents that have acted a witnesses by providing advice to buyer agent *b* regarding vendor agent *v*
W_b	Set of all advisor agents that have ever acted as witnesses by providing advice to buyer agent *b* regarding a selected vendor

Chapter 1

Introduction

1.1 Overview

When autonomous software agents are deployed in an open, decentralized multi-agent system for the purpose of conducting real e-commerce transactions, it is critical that they are equipped to reason and make clever decisions on behalf of humans. This challenge has given rise to a variety of areas of research that focus on topics such as how to design agents to intelligently socialize, negotiate, communicate securely and process financial transactions. Still, one key challenge continues to preoccupy researchers: how can autonomous software agents in a decentralized system reliably select other agents as partners in e-commerce transactions? In other words, how can autonomous agents determine whom to trust? As the author Dasgupta wrote: “Trust is central to all transactions” [1]. Without an effective and reliable way to judge the trustworthiness of potential trade partners, it is impossible to sufficiently manage the risk of the transaction and to therefore proceed with it. With agents

too apprehensive to engage in business, the viability of the e-commerce system at whole is at risk. And so, this issue is truly at the core of the design, development and deployment of any successful decentralized multi-agent e-commerce system.

Several obstacles face researchers who tackle this problem. Agents act completely independently and can therefore change behavior without notice. In open systems, agents are free to leave the system and return to abolish a bad reputation or to assume a new identity. In a decentralized system, there is no central authority and so agents must interact directly with one another and assess the credibility of others who offer opinions. Agents may have their own beliefs and attitudes that can make accurately sharing information between agents difficult.

Underlying all of these issues is the critical challenge in developing a sound trust model: trust is subjective [2]. The need to design a software model of a notion that varies from person to person and that is not clearly defined renders the problem extremely complex. Yet, especially given the significance of the problem, over the course of more than a decade many researchers have tried to take on the challenge and have proposed systems that model trust using a wide array of methods and parameters, for example [3,4,5,6,7,8,9,10,11,12] as well as [13,14,15,16,17,18,19,20,21,22] and [23,24,25,26,27,28,29,30,31,32].

Many of the proposed approaches include agent reputation as an integral part of the trust model, since "trust is based on reputation" [1]. Reputation, in turn, is based on past behavior observed and reported by (e.g., via word-of-mouth) other agents and is typically communicated between agents using a reputation rating. Agents who provide reputation ratings to other agents are known as witnesses or advisor agents. Reputation models define mechanisms for soliciting information from these advisor agents and for combining the

responses received with an agent's own past experience in order to make a trust judgment. The research presented in this thesis builds upon existing research in the area of reputation and trust modeling, aiming to introduce a new metric that can be used to enhance existing approaches, or to inspire new ones.

To find ways to improve decentralized reputation and trust modeling systems, over the course of this research some successful centralized reputation systems have been investigated and the data that is made available on those systems has been analyzed. This has revealed that successful centralized systems such as eBay present length of membership data as well as activity level indicators for sellers, buyers and advisors. In this thesis, the feasibility and effectiveness of applying equivalent parameters to decentralized reputation and trust models are investigated. As a result, it is proposed that reputation modeling in distributed e-commerce systems can be improved by adopting a new metric: *seniority*. This value represents a measure of an agent's participation in a multi-agent system, based on the agent's age and activity level. Through experimental evaluation, it is demonstrated that the seniority metric can be used to enhance reputation and trust modeling systems to reduce the risk and increase the success of agents who are new in an e-commerce marketplace and to produce better overall results for all agents in the system.

1.2 Motivation and Approach

Through the review of available literature about trust and reputation systems, the following common problems have emerged as being unresolved.

(i) A clear method for identifying reliable sources of reputation information when an agent first enters the e-commerce system is not defined.

(ii) Mechanisms for maintaining a credible set of agents as advisors are based on past experience in the e-commerce system, with accuracy increasing with the amount of experience. Therefore, new agents and agents who have a low frequency of interaction are vulnerable.

(iii) Approaches for weighting recommendations from various agents are dependent upon past experiences, and are therefore not useful for new agents and agents who use the e-commerce system infrequently.

(iv) Rating new agents fairly is largely unaddressed, and is critical since new agents need to be able to compete in an open e-commerce system.

(v) Robust measures for thwarting malicious agents who leverage the nature of open, distributed multi-agent systems through tactics such as identity changing require further development and additional approaches.

These problems are often peripheral to the core research presented since they lie mainly in the area of boundary conditions, such as the cases of: new agents faced with trust decisions; new agents as the subject of trust decisions; infrequent system users and malicious users. As a result, they are issues that persist, largely unresolved.

Another pattern has been observed through the review of related work in this field. Popular centralized reputation systems such as eBay, Epinions and others present information about a user's global participation in the system, including membership date and activity level. In contrast, none of the decentralized systems presented in the literature reviewed employ equivalent parameters.

Psychological research supports the importance of age, as indicated by membership date, and experience, as indicated by activity level, as indicators of influence that a member of a

community has over other members in the group. Experiments in [33] indicate that within human groups, leadership and organizational structure naturally emerge based on participants' seniority, which in turn is dependent on age, familiarity and experience.

Given this background, a new metric, *seniority*, has been developed to address the problems unique to agents who are new or have little experience in the e-commerce system (and therefore haven't any familiarity with regard to other agents). Based on an agent's age and activity level, seniority can be used to assess the credibility of unknown agents and therefore improve the success rate of purchasing agents, especially new agents and agents with low activity levels, in e-commerce systems.

In this thesis, the seniority metric is formalized then a trust model based on the seniority metric is presented and evaluated through experimental analysis.

1.3 Contributions

The research and results documented here make the following key contributions to the field of trust and reputation modeling:

(1) Identification of simple parameters that can be used by agents who are new to an e-commerce system to judge the credibility of potential advisor agents. By using the new seniority metric proposed here, novice agents are able to build a strong advisor list immediately and significantly reduce their risk, especially for the first few transactions in the e-commerce system.

(2) Provision of parameters to accurately weight advisor recommendations. By using the proposed seniority metric to weight ratings provided by advisors, all agents can benefit from reduced risk in the system and better results on an ongoing basis.

(3) Presentation of a simple metric that can be used to protect against malicious agents who intentionally exit and reenter the open multi-agent system under a new identity, attempting to reset their reputation then manipulate the opinions of others. By using a time-based measure of credibility like seniority, agents who reenter the system will effectively eliminate any of their credibility and therefore any influence they once had.

(4) Development of a metric that can be incorporated into virtually any existing trust model. While the seniority metric is isolated for analysis in the model presented in this research, its design is compatible with most of the existing trust models and can be used to enhance their effectiveness.

These contributions to the field of study were validated by the publishing of a paper with initial findings:

Catherine Cormier and Thomas Tran. Improving Trust and Reputation Modeling in E-Commerce Using Agent Lifetime and Transaction Count. *In Proceedings of the Fourth International Conference on E-Technologies (MCETECH-09),* pages 184-195, Springer, May 2009 (**Best Paper Award Nominee**)

The paper was presented at the MCETECH '09 conference and was selected as one of five nominees for the best paper award.

1.4 Thesis Outline

This thesis explores the viability of modeling the hierarchy of expertise that develops in a community as time passes and participants evolve. A new metric, seniority, is proposed as a means of quantifying an agent's level of expertise, and its effectiveness is evaluated through experimental analysis. This work is presented as follows:

Chapter 2: Reputation Systems in E-Commerce sets the stage for the research by introducing the concepts behind autonomous agents in multi-agent systems, the role that reputation systems play in modern e-commerce systems, and the general workflow of reputation models.

Chapter 3: Types of Reputation Systems discusses trust and reputation systems that are described in existing literature. It provides an analysis of the parameters that are currently employed in various reputation models and as well as the roles that experience, community structure and agent age play in these systems.

Chapter 4: Seniority to Measure Credibility in Reputation Systems describes the problems addressed and motivation for the research. The seniority metric is introduced and defined as a means to enhance reputation systems. Finally, the proposed approach is presented, explaining specifically how seniority can be used when selecting advisors, aggregating ratings and updating advisor lists.

Chapter 5: Experimental Evaluation describes the software simulation used to evaluate the effectiveness of the seniority metric and presents the experimental results obtained.

Chapter 6: Discussion offers a general discussion of the approach and results. More specifically, given the results obtained during the experimental phase, the suitability of seniority to reduce the ramp-up time for new agents and to improve the success rate of agents overall is presented.

Chapter 7: Conclusion closes the thesis with a summary of the advancements obtained and offers suggestions for future work in this area.

Chapter 2

Reputation Systems in E-Commerce

This thesis builds upon existing research in the broad area of trust and reputation systems for intelligent, autonomous software agents in open multi-agent systems. The following section provides a discussion of the common elements of trust and reputation systems: the key concepts, the motivation and the overall function of trust and reputation systems. In particular, a general description of software agents and multi-agent systems is presented. As well, an introduction to trust and reputation systems in e-commerce is provided, starting with definitions of trust and reputation and concluding with a functional overview of typical reputation systems.

2.1 Agents and Multi-Agent Systems

An ***agent*** is a software element that operates on a behalf of a user to make decisions for some purpose. As described by Garcia et al. [34], software agents are "viewed as complex

objects with an attitude." That is, they are like other software objects in many ways but have goals and knowledge. Further, agents are autonomous entities that can adapt to an environment or situation and can learn from experience. Amongst each other, agents can interact and collaborate. As such: "Agents must be social entities" [35]. It is the modeling of the social behaviors in software agents that presents an enormous challenge, and that fuels the extensive research that is performed in this field.

Systems in which agent interaction and collaboration may occur are ***multi-agent systems*** (MAS). Agents may interact in a variety of ways, including sharing information and engaging in transactions. Since software agents are representatives of their human users, in a general sense a multi-agent system may be considered a "virtual community," as used by Abdul-Rahman&Hailes [3]. Multi-agent systems are used for a wide variety of specific applications [34]; for example, as an electronic marketplace [7], as an education system [36], for the the exchange of Knowledge Objects [26] or for peer-to-peer file sharing [13,23,27,37,32].

In her article, *Multiagent Systems* [38], Katia Sycara identifies four characteristics of multi-agent systems:

(i) Limited agent view: relative to the whole system, each agent has access to a limited amount of information and problem-solving capabilities;

(ii) Decentralized control: there is no single point of global control;

(iii) Decentralized data: information is distributed in the system;

(iv) Asynchronous computation: functional logic is distributed and performed without coordination.

These properties heavily influence and define the constraints on the design of trust and reputation models for multi-agent systems.

The research presented within this thesis focuses on multi-agent systems used for e-commerce. This distinction is important because, for example, as Ruohomaa points out, e-commerce systems and peer-to-peer systems each have unique reputation modeling needs due to the inherent differences between the risks involved in their transactions [39]. However, the same principles shown here for e-commerce could be applied to MAS more generally. In e-commerce systems, the transactions may carry very high financial risk (e.g., purchase of electronics, jewelry or vehicles) or a considerable business risk (e.g., purchase of business mobile or Internet service). This risk must be mitigated by the user's ability to make informed, effective decisions about which agents to trust to proceed with transactions. If agents are not able to make accurate trust judgments and therefore fail to be successful in their transactions, their trust in the overall e-commerce system will falter, putting at stake the viability of the e-commerce system as a whole.

2.2 Trust and Reputation Systems

Much research in the area of trust in computer science has focused fundamentally on security, especially in the areas of cryptographic protocols and the use of certificates [3,40]. These approaches address serious issues such as ensuring that electronic communications cannot be intercepted by third-parties and authenticating the identities of other users. However, they do not consider if or how well another user will cooperate. According to Yu and Singh [40] "the trustworthiness of a correspondent is viewed as the expectation of cooperative behavior from that correspondent." This definition highlights a key attribute of trust—that it is tightly coupled to the notions of expectation and cooperation.

This expectation of cooperation can be very powerful. Our trust in another, as Dasgupta argues in [1], influences the actions we take, i.e., whether we choose to depend on the cooperation of another. As he states, trust is "correct expectations about the actions of other people that have a bearing on one's own choice of action when that action must be chosen before one can monitor the actions of others." In e-commerce, a buying agent must be very confident in its belief that a particular vendor will act as expected before it will proceed with the transaction.

Gambetta's following definition of trust is widely accepted in this area of research and presents the challenge that has given rise to many of the trust models developed:

> ...trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever to be able to monitor it) and in a context in which it affects [our] own action. [2]

From these definitions, three key attributes of trust can be identified:

(i) Trust involves an expectation of cooperation;

(ii) Trust influences our actions;

(iii) Trust is based on a subjective judgment.

These elements must all be considered when developing trust models for multi-agent systems. That is, trust in another agent must be considered before interacting with it. This trust must be based on the trusting agent's expectation that the trustee will cooperate in a beneficial manner. Finally, each agent must be able to judge the trustworthiness of another agent independently. Thus, for the purposes of this research we will define trust as follows:

Definition 1. Trust is the subjective judgment of expected cooperation from another agent such that it affects our own action.

To determine the trustworthiness of an agent, it is in many cases wise to consider that agent's ***reputation*** within the community. Reputation is the word-of-mouth proliferation of the agent's trustworthiness as perceived by other agents in the community. Dasgupta [1] states that, "...trust is based on reputation and that reputation has ultimately to be acquired through behavior over time in well-understood circumstances." This close relationship between trust and reputation as well as the notion that reputation is based on past behavior are concepts reinforced and expanded upon by other researchers.

Reputation is a concept that has been researched extensively in a number of contexts by a variety of researchers, such as economists, people who study scientometrics, computer scientists, evolutionary biologists and sociologists [41]. Economists have studied reputation from the point of view of game theory to explain irrational behavior of participants. Researchers in the field of scientometrics measure reputation based on the number of cross-citations obtained by an author or journal. Evolutionary biologists measure reputation as the "ratio of cooperation over defection" [41]. And, finally, sociologists consider reputation a network parameter in a society of agents [41]. These views share several common themes, particularly the idea that reputation is a measure of an entity.

In the field of computer science, a number of definitions of reputation have also been proposed. Yu and Singh [40], for example, state that reputation comprises "social mechanisms for incorporating the knowledge of other agents" when one agent judges the trustworthiness of another. This definition highlights a key attribute of reputation—it

depends upon a set of approaches. This characteristic of reputation is fundamental to this research, since it indicates that successful reputation modeling must be multi-faceted.

Further, echoing the point made in the general definition offered by Dasgupta, Abdul-Rahman and Hailes [3] indicate that reputation is heavily dependent upon past behavior: "A reputation is an expectation about an agent's behavior based on information about or observations of its past behavior." This definition introduces a critical point: the notion that the past behavior may be either observed first-hand or communicated by others (i.e., by "word-of-mouth"). This point will factor into heavily into the concept of reputation used in this research.

From these definitions, the following understanding of reputation may be derived:

> **Definition 2.** Reputation is a measure of trustworthiness based on the direct observation of an agent's past behavior, as well as referral information obtained from other agents' past experiences.

Reputation modeling is the design of approaches to reproduce the notion of reputation as it exists between humans in an abstract environment. More specifically, from a computer science perspective, "Reputation mechanisms are about generation, discovery and aggregation of rating information in electronic e-commerce systems" [28]. This highlights three key attributes of all reputation modeling systems. Reputation models must specify how reputation ratings are

- generated;
- communicated between agents; and
- combined with other parameters into a single value.

Wang and Singh [42] point out three key challenges to designing reputation systems for open environments which are the ability of the agent to:

- freely join or leave the environment;
- be uncooperative; and
- provide biased witness information, such as ratings and recommendations for other witnesses.

Several ***Reputation reporting systems***, as named by Mui, et al. [41], exist. These are centralized systems where users can report their rankings of other users. One example is the widely successful reputation tracking system included at eBay, in which sellers and buyers accumulate positive and negative ratings from users with whom they have engaged in transactions [43]. Other familiar examples of trust and reputation systems employed today include Amazon [44], Epinions.com [45] and the Apple App [46] store which provide venues for individuals to report their rating of products. While not specifically designed as trust systems, social media tools such as Twitter and Facebook have provided a peer-to-peer communication network that makes it easy for people, even complete strangers, to freely exchange opinions and rating information.

Since reputation reporting systems are centralized and require human interaction, they are not suitable for deployment in distributed multi-agent systems. However, especially given their widespread use and longevity, their successes and failures can serve as guides for the design of robust decentralized reputation systems.

2.2.1 Reputation System Operation

While reputation models can be designed to operate in any fashion, the majority of the systems proposed in the literature reviewed over the course of this research follow the same basic operating procedure, which is depicted in Figure 1.

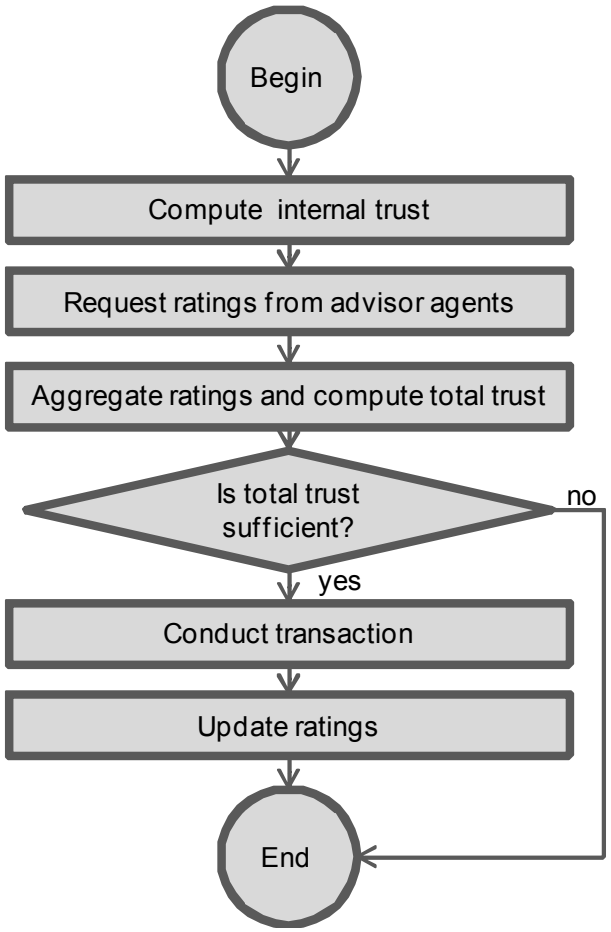

Figure 1: High-level steps followed by agents in most reputation models

To discuss the actual steps of operation further, consider an agent who is interested in buying a product or service from a vendor. These will be referred to as the buying agent *b* and the vendor agent *v*, respectively. In order to determine whether or not to proceed with the purchase from *v*, agent *b* performs the following steps:

1. ***Compute internal trust:*** Many of the reputation models documented in the literature draw on the notion of *internal trust* (which is also commonly known as *direct trust*). Internal trust is derived in some manner from agent *b's* prior transactions with vendor *v*. Thus, the first common step is to have *b* draw on its past direct experiences with *v* to calculate an internal trust value, denoted here as IT_b^v.
2. ***Request ratings from advisor agents***: In the second step, agent *b* solicits opinions about *v* from other agents in the MAS. This is the word of mouth stage where reputation information, based on other agents' experiences with *v*, is shared. Agents from whom feedback is requested and received are called *advisors, witnesses or recommenders*. Within this thesis, such an agent is referred to as an *advisor* agent and is denoted by *a*. The advisor rating provided by an advisor agent *a* to buying agent *b* regarding vendor *v* is denoted AR_a^v.
3. ***Aggregate ratings and compute total trust:*** Once the buying agent *b* has computed and collected the necessary information including, but not limited to, internal trust and ratings from advisor agents, it combines the values and computes a total trust value. While every model presents a unique approach to each of the steps in the process, it is the method employed in this step that is most unique between models. In the end, the objective of the systems is similar: to enable buying agent *b* to compute a value that represents its overall trust in vendor *v* given all of the information available. Here this value is referred to as the *total trust* that *b* has in *v*: T_b^v.
4. ***Make a buying decision based on total trust:*** Given the total trust computed in step 3, buying agent *b* must decide whether to engage in a transaction with *v*. The method

used is often as simple as: proceed if $T_b^v > \Theta$, where Θ represents some pre-determined trusting threshold.

5. ***Provide feedback:*** In this final step, based on the outcome of the transaction performed in step 4 (if it was indeed performed), agent *b* updates its records for future reference. Depending on the model, this may include whatever kind of feedback loop is required, e.g., storing a record of the transaction in an internal table, updating a trust value or revising credibility ratings for advisors.

By presenting these five common steps for reputation models, specific points at which improvements can be made have been identified, and a framework for defining an experimental trust framework has been defined.

With this common framework as background, Chapter 3 presents the distinctions in the various reputation models by discussing their approaches and analyzing the parameters that they employ. In Chapter 4, these high-level steps serve as the basis for the definition of the proposed seniority reputation model. Basing the model on this common framework simplifies the application of the principles investigated here to other models.

Chapter 3

Types of Reputation Systems

Given the importance of trust and reputation modeling in multi-agent systems and the inherent complexity of designing software agents that replicate human social, subjective behaviors, it is not surprising that this has been an area of intensive research for many years. Through this research, a wide variety of models have emerged. A good overview of these models is presented by Zhao et al. in [31] and a detailed analysis of seven of the most commonly referenced models is given by Ruohomaa et al. in [39].

In this section, differences in the approaches used by various models are highlighted through a discussion on types of reputation systems. First, the difference between centralized and decentralized models is presented and then models are analyzed based on the parameters they employ. Finally, reputation systems that use the notions of experience, community structure and aging are discussed.

3.1 Centralized and Decentralized Systems

The distinction of centralized versus decentralized reputation systems is used to indicate where reputation information is collected and stored.

In a centralized system, there is a single authority that maintains a record of all of the reputation rankings and associated information. When an agent is interested in obtaining the reputation information, it simply contacts the central authority. Several successful online marketplaces such as eBay, the Apple App Store, Epinions.com and Amazon offer centralized systems for reporting rating information [43,46,45,44]. At the eBay site, other users (buyers and sellers) are rated; at the Apple App Store, software applications available for purchase are rated; and at Amazon and Epinions.com consumer products are rated. In the eBay and Apple App Store systems, users generate ratings once they have engaged in a transaction; at Amazon and Epinions.com the rating may be contributed even without a prior transaction occurring. In all cases, users publish the ratings publicly at the site for discovery by other users. Rating aggregation is then left to the individual human user; each may interpret the ratings and other information about the user or application and make a trust decision in their own manner.

As described in Section 2.1, in multi-agent systems there is no central point of control and information and computing resources are distributed. Therefore, for a reputation system to function in an MAS environment, it must be a decentralized system. In these models, reputation information is distributed among the agents in the system. Typically, it is stored with the agents who are performing the trust evaluations; however, in NICE [15] the reputation information is stored with the agent who is to be evaluated. In either case, the

computational logic is distributed among agents, with the trusting agent equipped to make trust decisions autonomously.

A third category of multi-agent systems exists: semi-centralized models. In the models presented in [47] and [48] for example, there are one more agents responsible for managing the reputation information. Although these models can be deployed in distributed multi-agent systems, they require agents in charge of the reputation information which thus become small control centers. In this way, they are considered for, the purposes of this research, semi-centralized.

3.2 Parameters

All reputation models require some input into their system so that trust can be assessed. These parameters can be used at various phases in the function of the model: for example, during internal trust computation or in the aggregation of ratings. *Table 1: Parameters used in sample reputation models* provides a summary of the parameters used by a sample of twenty-one unique reputation models. While it shows that a variety of parameters have been employed, it is clear that the most of the models rely on internal trust and recommendations as their primary sources of information.

When internal trust is computed, that is a trust value stored by one agent regarding another agent, the following input is typically used:

- ***Transaction history:*** Since internal trust is based on the direct experience that one agent has had with another, it follows that the transaction history between two agents is the primary contributor to the computation of internal trust. The transaction history employed may be as detailed as a table of jobs performed including job ID, remote

Table 1: Parameters used in sample reputation models

Model	Parameters	Model	Parameters
Amazon [44]	• Ratings and written reviews	H-Trust [37]	• Local Service History Table (for each service locally observed: job ID, remote peer ID, date, service importance and service quality) • Score from recommenders
Beta Reputation System (BRS) [12]	• Number of positive feedback • Number of negative feedback • Forgetting factor	NICE [15]	• Internal rating • Ratings between agents along the path from agent A to B
Computation Distributed Reputation Model for B2C [49]	• Transaction ratings • Rater's credibility • Reputation lifetime • Transaction value • Number of malicious incidents	PeerTrust [27]	• Feedback from peers • Total number of transactions with other peers • Credibility factor • Transaction context factor • Community context factor
Con-resistant model [22]	• Conning pattern	PET [16]	• Internal rating • Interaction ratings • Number of interactions
DynamicTrust [50]	• Self-declared attribute ratings • Number failed interactions with rating and time record for each • Number successful interactions with rating and time record for each • Ratings from other agents	PowerTrust [32]	• Internal trust ratings • Number of feedback in the system • Number of nodes with the same feedback amount • Rating index (order of feedback amount in the decreasing list of feedback amounts)
eBay [43]	• Feedback score • % positive feedback • Member since • Number of feedback • Short comments • Colored star icon (based on feedback score)	Recommendation Algorithm for Knowledge Objects [26]	• Community position (self-declared), • Experience level (self-declared) • Adjustment value for experience level (controlled by manager agent)
Edge Creation Process [14]	• Indirect and direct ratings • Number of interactions • Time recency of last transaction	REGRET [21]	• Number of impressions for rating • Rating deviation
EigenTrust [13]	• Trust ratings from individual ratings	Travos [25]	• Number of successful interactions • Number of unsuccessful interactions
FIRE [11]	• Maximum number of evidence • Rules for role-based trust • Role-based trust reliability • Maximum number of witness ratings or maximum chain length • Ratings provided as references by the agent being rated	Using Fuzzy reasoning [10]	• Number of evaluations • Evaluation value
FuzzyTrust [24]	• Payment method and payment time • Goods quality and delivery time • Remote peer's trust score, transaction amount and transaction date	α-Rank [4]	• Distance between agent nodes
Google PageRank [51]	• Number of inbound links • Score of inbound links		

peer ID, date, service importance and service quality as in [37] or it may be stored as a single cumulative rating that is adjusted after each transaction [52]. A single transaction rating may be general (i.e. satisfaction with overall transaction) or broken-down into specific attributes of quality of service.

- ***Number of transactions on which ratings are based***: In many models, the number of interactions on which a given rating is based acts as an input to computing internal trust, and is essentially used to judge the reliability of the internal trust rating. It may be provided either as an explicit total number of transactions as in [14]; or as an implied number of transactions based on number of records stored as in [37]. In the FIRE model [11] it is argued that there is a maximum number of transactions required to generate an accurate rating. Therefore, in FIRE the number of transactions on which ratings are based never exceeds the predefined local rating history size.

In the reputation models surveyed, there are also a common set of parameters used to compute the overall reputation of the evaluated agent, as it perceived by other advisor agents in the community. This calculation is most often based on:

- ***Rating values from advisor agents:*** In the majority of the models presented in *Table 1*, ratings provided by other agents are used in the overall trust evaluation. In centralized systems like eBay [43], Amazon [44], Epinions.com [45] and the Apple App Store [46], these opinions are posted publicly via a centralized venue. At Epinions.com, a single aggregated rating is presented, while the others also provide the distribution of ratings (e.g., the number of 1, 2, 3, 4 and 5 star ratings). In each of these centralized systems, individual ratings are only published with a textual review.

Since written reviews are mandatory at Epinions.com, in their model every individual rating is available publicly.

In the decentralized models examined, recommendations are either global or collected and aggregated privately. In EigenTrust [13], FuzzyTrust [24] and PowerTrust [32], for example, a global trust value is computed for each peer and the values are stored in a distributed manner. And so, in these models, all agents are effectively advisors to all other agents. In most distributed models, such as [49], [9], [14], [11], [37], [27] and others, the recommendations from other agents are solicited and received at a personal level. That is, the agent making the trust decision asks for advice through some established protocol and receives a rating as a response.

- ***Credibility rating for each advisor:*** It is generally accepted by the models that employ the use of recommendations that advice is only as reliable as its source. Therefore, these models include the ability to judge and act upon the credibility of other agents who provide recommendations. The approaches and the input used in this process vary substantially.

  eBay, Amazon and Epinions.com provide links to the public profiles of people providing reviews so that the human users of these sites can evaluate the trustworthiness of the reviewer in question. The screen captures of Epinions.com in Figure 2 show precisely how this credibility information is conveyed to the user.

  Here, a link to the advisor (a person posting a product review) is included with the reviews on the main product page, as seen in Figure 2 (a). The corresponding profile page provides the date the reviewer joined the site, the number of reviews written,

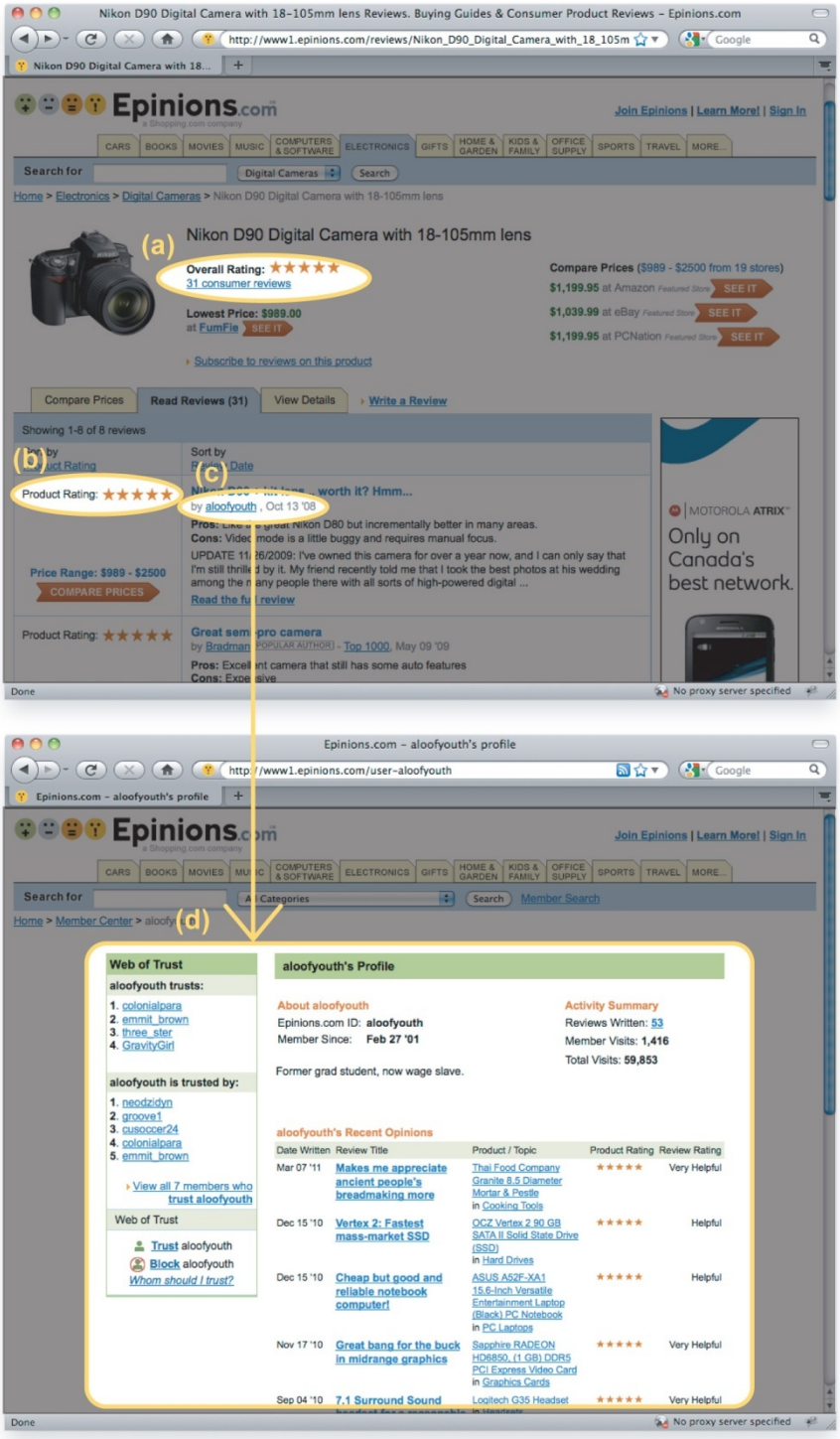

Figure 2: Sample product rating and advisor credibility information at Epinions.com

(a) Overall product rating [53]; (b) Exact product rating from single advisor; (c) Link to advisor's profile page; (d) Advisor's profile [54]

the total number of unique views to of all the reviewer's postings, links to all of their reviews and details about their "Web of Trust." This web of trust allows users to control their node in a network of trust relationships and is similar to the approach used in α-Rank [4], where trust is binary, and an edge in the trust network exists only when trust exists between nodes. Together, all of this information is presented to human users of the site so that they may accurately judge the credibility of the reviewer and consequently decide whether or not to trust their advice

In a similar fashion, eBay offers a link to the profile of a user who rates another user. Consider the screen capture sequence given in Figure 3. In the first screen, the product offering page, the overall rating for the seller is presented as well as a link to the seller's full profile. On the full profile, shown in Figure 3(b), further details of the overall rating are provided (e.g., the rating distribution) in addition to individual ratings with written reviews. To establish the credibility of each of these reviews, a link to the profile of the reviewer (or advisor) is provided. Unlike Epinions.com, where members participate exclusively to publish reviews, at eBay users can be any combination of seller, buyer and advisor. Therefore, the profile is not specifically about the reviews that they have posted. The eBay profile includes: the number of ratings the user has received (where the maximum number of reviews is the total number of transactions in which the user has participated), positive feedback percentage for the last year, the date that the user joined eBay, rating distribution, attribute ratings and individual ratings with written reviews.

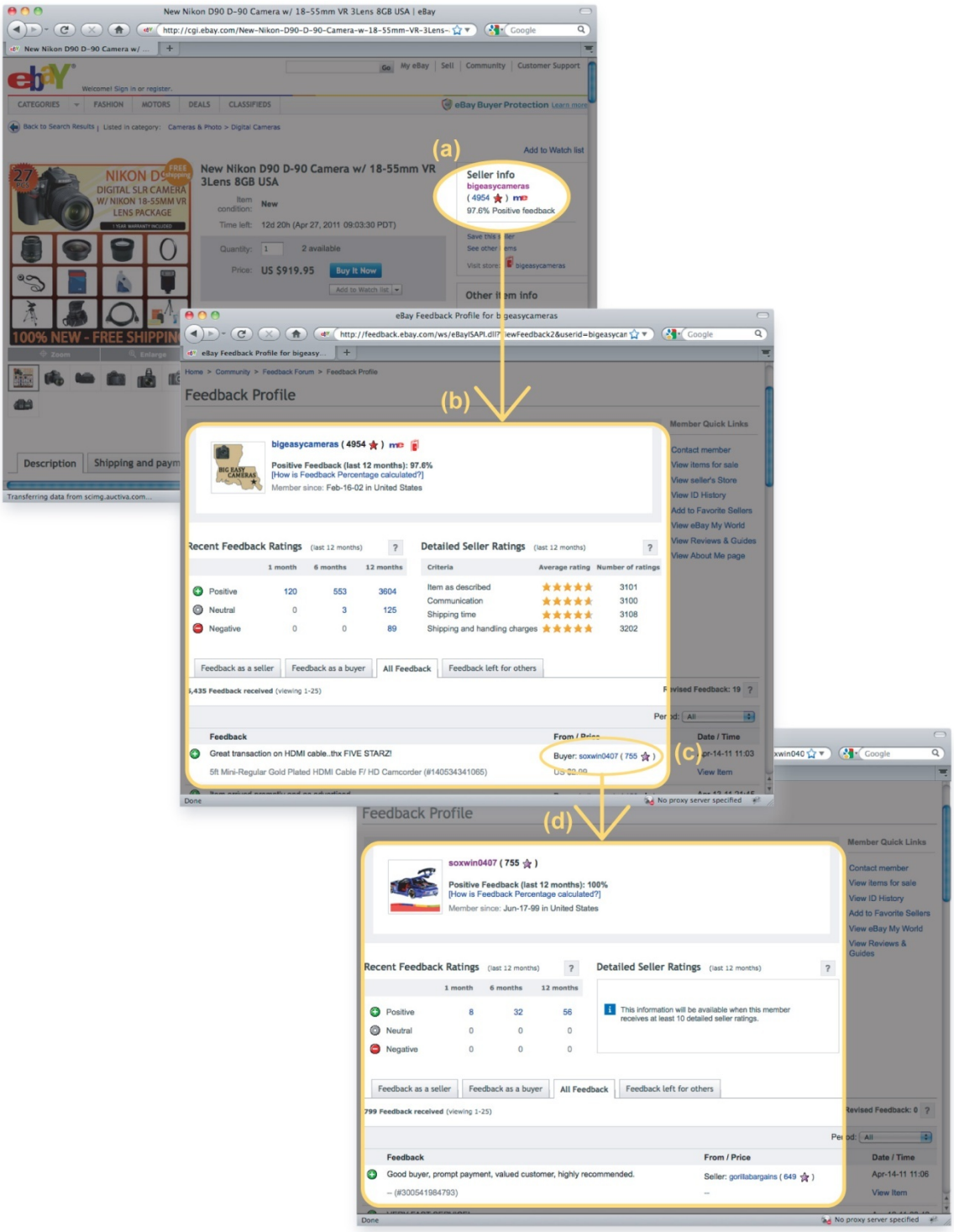

Figure 3: Sample seller rating and advisor credibility information at eBay

(a) Overall seller rating [55]; (b) Seller profile with exact rating from advisor [56]; (c) Link to advisor's profile page; (d) Advisor's profile [57]

In turn, these reviews have links to their authors and so the human user is free to explore a tree of profiles. It is the combination of all of these elements on the eBay profile page that enable the human user to decide the credibility of the ratings that they are considering.

Just as the set of parameters used to determine the credibility of advisors varies among the centralized reputation models, there are a number of different criteria used to judge advisor credibility in the decentralized models. For example, in [50], the internal trust value held regarding the advisor as well as the distance to the advisor in the network are used. The philosophy employed in [14] is as follows: trust ratings should contribute more as the number of interactions increases and should contribute less as the time since the last interaction increases. Therefore, in this model the authors propose using the number of interactions on which the rating is based as well as the how recently the interactions have occurred when judging the value of the advisor's rating. Among the parameters used in FIRE [11] to judge a witness' reliability are rules based on the roles that the agent and its advisor play with regard to one another. Finally, in H-Trust [37], a *local credibility table* is maintained by each agent to track and update credibility ratings assigned to advisors with which it has interacted in the past. It is based on these past interactions that agents are able to assess the trustworthiness of their advisors.

Through the analysis of the centralized and decentralized reputation models, a clear difference in the two approaches has emerged: the number of parameters used is higher in the centralized systems, where human users make the ultimate trust decisions, than in the decentralized systems where software agents make the trust decisions. The motivation for

keeping the criteria used low in the MAS implementations is to keep the load on the system and agents minimal. Still, given that the objective of the decentralized reputation systems is to model in agents the social behaviors of humans and that the centralized systems enjoy wide-spread acceptance and success, it seems clear that further research into how the principles of the centralized systems, especially the parameters that they use, factor into trust making decisions merits further investigation.

3.3 Aging Factor in Reputation Systems

As discussed in *A Trust Model Considering the Aspects of Time* [58], most reputation models do not take into account the effects of aging when evaluating its trustworthiness. When an aging factor is considered, it is the age of the evidence that one agent has about another and not the age of the agents themselves.

In some models, as the authors claim in [58], an aging factor is applied to weight the available evidence based on the relative age of each interaction. Unfortunately, when this approach is used, "aging" does not occur until a new piece of evidence is added. And so, if no newer evidence is obtained, the existing evidence will carry the same weight (or have the same "age") indefinitely.

In contrast, in their model, Spitz and Tüchelmann propose an approach that will account for aging based on time instead of on evidence count [58]. To do so, they propose a technique for factoring aging and inactivity by putting ratings into time slots. The lack of any ratings in a timeslot indicates a period of inactivity. This information can then be used when aggregating the ratings.

In another model, authors Khosravifar et al. propose the use of a parameter they call *time recency* when computing direct trust and indirect trust [14]. This value represents the age of

the most recent available interaction on record, either internally or from an external source. The model then ignores any ratings based on evidence with a time recency value less than a specified threshold.

All three of these approaches recognize that time and age are important factors in the relevancy of associated reputation and trust information. They all, however, use the principle that evidence becomes less relevant over time. This is a fair assertion, and certainly the experimental results presented by these authors support that point of view.

The research presented in this thesis proposes a look at the flip-side of age: that with age and experience comes wisdom. In other words, agents that have longevity in a system may provide more reliable reference information than agents who are newer to the system. While it is possible that older agents may leverage their experience to develop tactics for cheating the system, provisions such as ongoing credibility assessments can be employed to defend against such malicious behavior.

In this research, the effect of considering the age of the agent within the system rather than the age of the agent's most recent transaction is investigated.

3.4 Community Structure in Reputation Systems

In every community of individuals, there is a community structure that forms, explicitly or otherwise. In every group, there are experts and beginners, fanatics and dabblers, or leaders and followers. Yet, very few reputation models consider this attribute of the environments in which they are designed to operate.

In their paper *A Recommendation Algorithm for Knowledge Objects based on a trust model* [26], Vizcaino et al. present a unique means of expressing the structure of a virtual community. Each agent, upon entering the community, must self-declare the following

attributes: community position and level of expertise, where community position is the agents' level in the organization and the level of expertise is one of: beginner, competent, expert or master. The values are moderated by manager agents. In this model, when buyer *b* evaluates vendor *v*, it will first check the amount of direct experience it has. If it is insufficient to make a reliable trust evaluation, *b* examines *v*'s community position and level of expertise. Agent *b* trusts *v* only if the combination of these two values gives *v* a higher placement in the community social hierarchy than its own; this is referred to by the authors as *using intuition*. This is essentially a ***role-based*** approach since the community position and level of expertise are used to establish an agent's role in the community and based on that role other agents are able to define rules about how to trust it. This approach is distinct in the set of models assessed since:

(i) The trust assessment is based on some global properties of the agent rather than strictly on attributes defining the relationship between two agents (e.g., number of transactions or distance apart in the network). As a consequence, it is possible to make a trust decision about another agent without first having direct interaction with them. Therefore, this is beneficial to new agents who enter the system and don't have any past experience to draw upon.

(ii) Attributes are self-declared. This could significantly limit the suitability of the model for real-world use. Despite the periodic assessment of the values by management agents, there are opportunities for malicious agents to exploit this weakness.

(iii) By classifying agents according to their community position and level of expertise, the structure of the community begins to define itself, and expert agents become apparent.

Another model that seeks to detect the inherent hierarchy of the nodes in the MAS is given in FuzzyTrust [24]. In it, agents who frequently engage in numerous transactions are called *super users*. As a part of this research, Song et al. performed an analysis of eBay transactions and discovered the following: "The time interval between two adjacent transactions is short and stable for super users. For small users, however, the transaction intervals are long and unstable" [24], and "…super users conducted more than 10,000 transactions in three months, whereas the small users conducted only a few tens of transactions in several years" [24]. This shows that super users have a considerably greater amount of past experience that can be used to generate very useful advice for other agents. Small agents, on the other hands, have a desperate need for reliable advice since they each have very little experience to draw upon. Thus, there is a strong motivation for Song et al.'s approach: to identify the super-users and to ensure that their ratings have greater influence over the global rating derived using this model.

This is the same motivation that drives the use of community structure in the approach proposed in this thesis. In the approach presented here, agent attributes with global scope are used to define a hierarchy of seniority within the community. From there, agents are able to identify other agents who have high seniority and subsequently solicit advice from them.

Chapter 4

Seniority to Measure Credibility in Reputation Systems

While the models presented in the related research offer a variety of approaches for computing and applying internal trust as well as distributing, collecting and aggregating reputation information, several deficiencies exist. In particular, few models define the selection of an initial set of advisor agents to contact for reputation information. Additionally, in the majority of the models reviewed, a large amount of experience with the agents being rated and the advisors providing reputation information is required in order to be effective at judging the trustworthiness of others. These problems present significant challenges to agents who are new to the multi-agent-system.

To address these issues, a new metric, *seniority,* is proposed. This chapter explains in detail the motivation for this new metric, defines seniority and its computation, then

provides an example reputation model that employs the seniority metric for the selection of advisor agents and the aggregation of reputation information collected from these agents.

4.1 The Problem

Existing models for trust and reputation management in e-commerce systems fail to address or inadequately address issues that are specific to new agents and agents who use the system infrequently. And yet, these usage scenarios are real-world cases. In their research, as discussed in Chapter 3, Song et al. showed that only a few users in eBay are super users, some of whom conduct more than 10,000 transactions in three months, while many are small users who engage in only a few tens of transactions over several years [24]. Given that the majority of the users are small users, they cannot be ignored. Therefore, even without much or any experience in the system, agents must be able to make accurate trust decisions and yield positive results.

Furthermore, all agents in the system need to be equipped with tactics to defend against devious agents who use identity changing and ballot stuffing to deceive other agents. While some models in the literature do investigate measures to address these issues, further techniques are required.

The following subsections explain these issues further by identifying and explaining the five specific problems that exist with today's trust and reputation systems and that the proposed seniority model aims to address.

4.1.1 Initializing the Advisor List

In any reputation system that relies on advice from other agents, it is necessary to employ some method of identifying agents within the multi-agent-system from whom advice should be solicited. Some of the models such as PowerTrust [32] and FuzzyTrust [24] present a

means of computing a global trust rating, combining the opinions of all agents in the MAS. Thus, in these designs, the advisor list is effectively the whole set of agents in the MAS. In other models, a list of witness or advisor agents is maintained by each individual agent. However, very little information on initializing this list is provided in the literature. Without a specific and effective means of selecting an initial set of advisor agents, there is a high risk that when a new agent enters the system it will seek recommendations from untrustworthy or unknowledgeable agents.

To illustrate this point, consider a MAS employed for e-commerce. When an agent first enters the e-commerce system, it has no prior direct experience with any other agents in the system. Therefore, to select commerce partners it must rely solely on the recommendations of other agents. If an agent in the e-commerce system must perform several transactions before it can establish a credible list of advisor agents, then it must be willing to assume an extremely high degree of risk during its first several transactions. This is especially true if the transactions carry a very high value (e.g., electronics, automobiles or critical services).

In order to mitigate the risk to agents who are new to the e-commerce system, it is critical to devise a means of initializing the advisor agent list with a credible set of advisor agents. The primary challenge in this objective is determining a method to accurately assess credibility without any prior direct interaction.

4.1.2 Reducing the Experience Required

When the approach in a witness-based system is to use a list of advisors, the credibility of advisors is typically evaluated on an on-going basis and advisors may eventually be replaced over time. Therefore, as an agent gains experience by collaborating with others and collecting recommendations from its advisors, the influence of the agent's initial set of

advisors should diminish, and the agent's advisor set should become increasingly reliable. While in the long term this approach protects agents who are very active in the MAS, it leaves those who have a relatively low level of activity in the MAS extremely vulnerable.

In an e-commerce system for booking hotel accommodations, for example, there is likely a combination of agents who engage in transactions very regularly as well as agents who book only occasionally. In this case, it is not reasonable to expect an agent to commit to booking and staying in several hotels before it can accurately judge the credibility of its advisors and potential advisors. Instead, infrequent travelers must be able to effectively identify expert agents who can provide credible information about which hotels to book.

Therefore, in a more general sense, it is necessary to develop a means of enabling agents who engage relatively infrequently in transactions within the system to accurately select highly experienced, credible agents as its advisors.

4.1.3 Evaluating the Credibility of Advisor Agents

Even when the most highly credible agents are selected as advisors, there is a range of reliability in the recommendations that they provide. To model this spectrum of credibility, it is reasonable to weight the ratings received from the advisor list based on the degree of trust that the agent has in each individual advisor. In most models, such as [37], [27], [16] and [21] that employ such a technique, past direct experience with the advisor, or with other agents along the path that connects the agent to the advisor, is required. For agents who are new to the system or who have a low activity level within the system, the direct experience on which such judgments are to be made simply does not exist, and therefore the existing approaches are not suitable to them.

Returning to the example of an e-commerce system for booking hotel accommodations, consider a buying agent *b* who wishes to book a hotel room through the system. If *b* uses the system very infrequently, say once per year to book a hotel room for a family vacation, it will have a very limited transaction history to draw upon, and many of the transactions in the history will be out of date. Therefore, in this scenario a model that computes a credibility weight for advisor *a* based on past advice provided by *a* compared with past direct experiences by *b* will not be effective. Instead, *b* needs to be assured that it can weight the advice from its set of advisors accurately, without relying on past experiences, so that it can maximize the probability of booking a satisfactory family vacation.

To properly assess the reliability of ratings as they are received by advisor agents and to adjust the influence of the ratings based on this reliability assessment, it is necessary to develop a weighting and aggregation scheme based on publicly available information rather than solely on past direct experience.

4.1.4 Rating Novice Agents Fairly

In order to ensure that vendors who are new to the e-commerce MAS have an opportunity to be selected by buyers, build up a reputation and generally thrive, it is critical that the reputation model employed enables fair rating of such agents. Some models such as [59] are based on the premise that a good reputation should be difficult to develop but easy to lose. While this reasoning is sound, its application can potentially be problematic in the case of rating new agents. For example, if the rating is represented by a single value (e.g., 0 to 10) it would be impossible to tell if an agent with a rating of 4 had: (a) been historically providing reasonably good service, but recently service had declined significantly; (b) been providing a poor service regularly over an extended period of time; or (c) recently joined the system

and had provided an excellent service to very few individuals. If buying agents believe that such a rating is due to scenarios (a) or (b), new vendor agents could be starved of business.

This case has been identified and addressed by some models. For example, in [26] the authors propose using self-declared values of expertise and community position to enable the evaluation of the trustworthiness of a new agent in the system. The principle problem with this approach is that to determine the trustworthiness of an agent it is necessary to first trust the information that they have declared. Without some outside information, this quickly becomes a circular problem.

Consider again the hotel booking e-commerce system. Imagine that the system employs a five-star public rating system where stars are difficult to earn but easy to lose. A new vendor *v*, who represents a privately owned inn offering exceptional comforts and customer service, decides to make the inn available in the e-commerce site. Because it has been booked only a few times through the system, when buyer *b* seeks reputation ratings about it, the aggregated rating is a very low star rating, only 2 out of 5 stars. When buying agent *b* sees the low rating, it will likely reject the inn as a candidate and will instead book with another hotel. However, if the rating is put into context by showing, for example the number of bookings with the hotel and when the hotel became available via the system, *b* would be able to make a more informed decision and may opt to book with the new inn after all.

Distinguishing between service providers who have been providing a high quality of service for a short period of time versus service providers who have been providing a very low quality of service for a long period of time is very important in a successful reputation model.

4.1.5 Minimizing the Influence of Malicious Agents

There are a number of different ways in which dishonest agents can attempt to defraud other agents in the MAS. In an open MAS, one approach of particular concern is exploiting the agent's ability to freely exit or enter the system. That is, agents can simply leave the system when a poor reputation has been developed or when corrupt behavior has been detected then reenter under a new identity. This tactic may be described as *identity changing*. Deceitful agents may also collaborate with one-another. For example, agents may agree to provide false positive information about a friend and false negative information about a competitor. In systems where a rating must be supported by a number of interactions, *ballot stuffing* may be used, where an agent enters into a high number of free or almost-free transactions with one or more agents solely to inflate its overall reputation.

For this example, consider an electronic marketplace for consumer goods, deployed as an MAS. In it, malicious vendor *v* wishes to sell some low quality electronics, passing them off to buyers as high-quality goods. Advisor agent *a* is a friend of *v* who is willing to assist in the fraudulent behavior. Agents *v* and *a* quickly engage in a series of transactions so that *a* has a set of evidence on which to base its recommendations to other agents (note, in some systems it may even be possible to simply lie about the number of evidence). When buying, agent *b* solicits a recommendation rating from *a* regarding *v* and *a* responds with an artificially high rating supported by its set of transactions with *b.* As time passes and advisor agent *a* loses its credibility and therefore its influence on buying agents, it simply performs an identity change. In other words, agent *a* leaves the system, reenters under identity *a′* and repeats the procedure in order to reestablish credibility among the buying agents.

In order to minimize the effects of identity change and ballot stuffing, it is necessary to devise a scheme that makes it difficult and lengthy to build up credibility as an advisor in the system. One approach to this problem is to introduce parameters into the judgment of credibility that are based on time and whose values are outside of the control of the agents themselves.

4.2 Seniority as a Metric for Reputation Systems

In order to solve the problems described, this research investigates the feasibility and effectiveness of developing a new metric, coined *seniority*, which can be used to judge the credibility of agents without any prior exposure to the MAS. The following section defines the seniority metric, explains the motivation for its development and describes its components and computation in detail.

4.2.1 What is Seniority?

Seniority is, according to the Oxford Dictionary of English, "the fact or state of being older or higher in rank or status than someone else" [60]. From this definition, it can be derived that seniority defines a social ranking of individuals based on their age or some other criteria and that given the seniority of a set of individuals they can be sorted from least senior to most senior.

In 1982, researchers Insko et. al. reported on the small-group research they had performed to determine the effects of social familiarity and experience on seniority. They present the following definition:

> A seniority effect can be defined as a tendency for group members with longer tenure in the group to be relatively more influential in the determination of group-related activities. Typically the group members with

longer group tenure will be chronologically older, but this need not be the case. [33]

This definition highlights the important correlation between seniority and influence within a group. An individual with higher seniority typical has greater influence in the community. The research by Insko et al. showed that this was the case, even without rules to impose such allocation of power. That is, senior members of the group simply emerged as leaders [33].

From these definitions and observations, the following definition of seniority used in the context of this research can be presented:

> **Definition 3.** Seniority is the ranking of influence within a group, based upon one's age and experience.

The following sections will justify the use of seniority in distributed reputations systems, will formalize this definition mathematically and will describe a simple model that makes use of seniority as a metric for judging advisor credibility.

4.2.2 Why Seniority?

The inspiration for using seniority to enhance reputation systems comes from the intuitive notion that, in any setting, one should aim to select the wisest set of individuals as one's advisors and that—as the cliché goes—with age and experience comes wisdom. It seems natural that an agent in a multi-agent system would seek out recommendations from experts, where experts are other agents who have spent a lot of time actively using the system. For example, in a restaurant reservation system, a buying agent *b* would prefer to receive restaurant recommendations from a true local restaurant guru (i.e., someone who had frequented a large number of local restaurants over the course of many years), over the opinion of someone new in town who had only been to one or two local restaurants. And,

given advice from the two, *b* would assign greater value to the guru's opinion than that of the newcomer. To address the problem of assessing credibility without any prior knowledge of the MAS, the concept of using seniority has been conceived.

This approach is supported by the research presented in [33], which observed the behavior of human subjects through experimental analysis to identify the contributing effects of social familiarity and experience on seniority. In their experiments, small groups of three were asked to perform a task for seven cycles, but given no direction regarding the organizational structure of their groups. At the end of each of these cycles, individuals were asked on a questionnaire to identify the leader in their group (if one existed), then one member was replaced with a new one. In the first run, to test the effect of social familiarity on seniority, half of the groups had rotating members and half had stationary members (with the exception of the one individual replacement). In the second run, to test the effect of experience on seniority, half of the groups were comprised of inexperienced individuals, while half were made up of experienced people. Their results showed clearly that leaders naturally emerged based on their relative seniority in the group. In addition, their results revealed the following: (i) seniority increases with each generation (with age); (ii) social familiarity and experience both affect seniority, where age plays a bigger role when familiarity exists and a lesser role in a group of experienced individuals; and (iii) experience has a greater effect on seniority than social familiarity.

This research into the role of seniority within human individuals can serve as a guideline for developing software models for multi-agent systems. When identifying leaders in a group, or advisors in the case of reputation models, the three confirmed factors of seniority should be considered: age, familiarity and experience. In many ways, some of the existing

models already consider familiarity. For example, in [4] trustworthiness is based on the distance in the trust network between the trustor and the trustee; this may be deemed the degree of familiarity. However, when an agent is new to the system or is relatively inactive in the system, there is no social familiarity to draw upon. Therefore, to address the unique problems presented by new and inexperienced agents, as described in Section 4.1, it is necessary to take into account the other two factors of seniority: age and experience.

The use of these parameters is further reinforced by the presence of equivalent values in the successful centralized systems used by eBay [56] and Epinions.com [54]. As an indication of member age, both of these sites provide the date that a member in question joined the service. With regard to experience in the system, at eBay, the total number of feedback given for a member is presented. Since one feedback can be given for each transaction, it follows that this number is less than or equal to the total number of transactions in which the member has been engaged. At Epinions.com, the member's experience is summarized by providing the total number of reviews written, the number of views by other members and the total number of views of all of the member's reviews (including views by members and non-members). It is left to the human user to determine how to make use of these pieces of information when making a trust decision. Given the widespread success of these two reputation systems, the presence of age and experience measures at both of these sites is a strong indication that equivalent information should be employed in decentralized systems.

4.2.3 Expected Benefits of Seniority

It is anticipated that a model that properly employs seniority will be able to address the issues defined in section 4.1 as follows:

(i) ***Initializing the advisor list:*** Because seniority is based on the notions of age and experience within the system, ranking can occur without any prior knowledge of other agents or experience within the system. Therefore, new agents will be able to enter the MAS, immediately identify the senior agents and initialize their advisor list with agents. As a result, it is expected that by implementing seniority, new agents will be able to make successful trust decisions beginning with their first transactions in the system.

(ii) ***Reducing the experience required:*** On an ongoing basis, agents who do not use the system frequently will be able to turn to senior agents in the system who have a long and active history for advice. Therefore, agents who use the system only occasionally can benefit from the experience of others and consequently can yield high results from the few transactions in which they do engage.

(iii) ***Evaluating the credibility of agents:*** Since seniority is a ranking, where some agents have more seniority than others, this metric can be used to weight advisor recommendations. This will give agents with more seniority greater influence than those with less seniority, resulting in more accurate aggregation of recommendations.

(iv) ***Rating novice agents fairly:*** Because an agent's seniority is always known, the possible ambiguity of ratings is eliminated. It will always be possible to distinguish the agent with a mediocre score due to poor performance from an agent whose score is mediocre because they are relatively new and in the process of ramping up their reputation.

(v) ***Minimizing the influence of malicious identity changers:*** Given that seniority is based largely on age, an agent who elects to change identities will eliminate any seniority, and therefore influence, that they once had. Because age increases only with time, there is no way for the deceitful agent to fast-track its seniority development. Even if the agent participates in ballot stuffing techniques, only the experience factor will increase and the overall effect will be moderated by the young age of the new identity.

Seniority, the use of age and experience as its parameters and its use in finding and maintaining advisors, can be formalized as follows.

4.3 Parameters of Seniority

As the preceding section has established, there are two contributing factors to consider in developing a seniority-based ranking scheme: the amount of time that an entity has been a part of the organizational structure and the number of contributions that the entity has made. In the context of an e-commerce system, these factors can be formalized as the following two parameters: agent lifetime and transaction count.

4.3.1 Agent Lifetime

Agent lifetime is the amount of time that an agent has been a part of the multi-agent system, or, more simply, the agent's age within the system. Malicious agents who have developed a negative reputation may be tempted to exit the e-commerce system and reenter to perform an identity change [61]. Conversely, agents who have established a positive reputation over time are motivated to stay within the system. As a result, an agent's lifetime can be used as an indicator of trustworthiness.

Agent lifetime is calculated at any given time *t* using a *timestamp* assigned to the agent when it enters the multi-agent system. For any agent *a*, the agent lifetime is given by:

$$l_a(t) = 1 - \frac{\tau_a}{t} \quad (1)$$

Where: $l_a(t)$ is the agent's lifetime at time *t*; τ_a is the timestamp from the time at which the agent entered the MAS[1]; and *t* is the timestamp for the point in time in question.

Using this formula, agent lifetime is normalized so that $l_a(t)$ is always in the continuous range [0,1]. Agents that have been in the system for a long time will have a lifetime approaching 1, while agents that have recently entered the system will have a lifetime close to 0.

4.3.2 Transaction Count

An agent's transaction count is the total number of transactions in which the agent has participated over the course of its lifetime. This is a measure of the agent's activity level within the system. It is presumed that an agent that has participated in a large number of transactions is more experienced than an agent that has participated in fewer transactions. And, by extension, the advice provided by the agent with a higher transaction count is more valuable. As discussed previously, exceptions may exist where, for example, malicious agents engage in a large number of low-value transactions in order to falsely inflate their transaction count and to use ballot-stuffing techniques to falsely inflate or deflate another agent's reputation. However, the use of agent lifetime in conjunction transaction count may be useful in detecting such malicious behavior.[2]

[1] In this research, standard universal time is used for the agent's time stamp and current time, and these are represented by standard time units.

[2] The precise protocols for securely communicating the agent lifetime and transaction count values are beyond the scope of this research, which aims to prove the value of the parameters in reputation systems. Designing these protocols is deferred to future work as described in Chapter 7.

The transaction count for any agent *a* at time *t* is denoted by $n_a(t)$ and is automatically incremented whenever a transaction occurs between agent *a* and any other agent.

4.3.3 Computing Seniority

In the proposed approach, an agent who has been an active participant of the multi-agent system for a relatively long time, as compared with the other agents in the system, is considered a senior agent within the system. By identifying agents who are the most senior in the system, agents entering the system will be able to establish an initial set of advisors whose recommendations are based on as much experience as possible. Further, agents' seniority can be used to weight the advice received from advisor agents so that advice based on more experience more heavily impacts the overall reputation calculated.

The *seniority* of agent *a* at any time *t* is given by the product of its lifetime and transaction count:

$$s_a(t) = l_a(t) \cdot n_a(t) \qquad (2)$$

Where:

$s_a(t)$ is the computed seniority of agent a at a given time t;

$n_a(t)$ is the total number of transactions in which agent a has engaged over its lifetime in the e-commerce system[3], as described in section 4.3.2; and

$l_a(t)$ is the lifetime of agent *a* at time *t*, as computed using equation (1).

4.3.4 Seniority Ranking

As given by the definition of seniority in section 4.2.1, seniority provides a ranking of influence within a community. Therefore, the following rules can be established:

[3] Here the total transaction count is used in the seniority computation, and seniority is thereby a number of transactions weighted by the agent's age. Alternatively a transaction threshold may be employed, and this approach may be investigated in future work.

1. An agent *a* is said to be *more senior* than agent *b* if $s_a(t) > s_b(t)$.
2. Any agent *a* that is new to the system or has never participated in a transaction is considered a *novice agent* and has a negligible seniority value: $s_a(t) \cong 0$.

These rules are useful for comparing the seniority of agents to select advisor agents, for example, or for detecting special cases that pertain to novice agents, such as determining the rating of a novice agent.

4.4 The Seniority Reputation Model

To illustrate how seniority can be used in a reputation system, this section presents a model that employs seniority in the selection of advisor agents, in the weighting of advisor feedback during rating aggregation, and in the revision of an advisor list.

The model described here supposes a marketplace *M* which is an open MAS employed for e-commerce. For simplicity, it is assumed that the marketplace is used for buying and selling items in a single context. In other words, all vendors in the marketplace offer similar products and all buyers seek that single type of product. While it is believed that seniority can be applied to models that account for various buying contexts, the objective of this model is to prove the merit of the seniority metric independently of other factors in a reputation model.

Marketplace *M* is populated with buyer agents and vendor agents, represented by *B* and *V*, respectively. Buyer agents are free to buy from any of the vendor agents in the marketplace. More formally, the environment for this model can be expressed as follows:

$$B = \{b_1, b_2, b_3, \ldots, b_n\} \quad (3)$$

$$V = \{v_1, v_2, v_3, \ldots, v_n\} \quad (4)$$

$$M = B \cup V \quad (5)$$

Where B is the set of buyer agents, V is the set of vendor agents and M is the e-commerce marketplace. Additionally, all buyer agents can provide advice to other buyer agents; therefore, every buyer agent is also a potential advisor agent. This is expressed as:

$$ADV = \{adv_1, adv_2, adv_3, \ldots, adv_n\} \quad (6)$$

$$ADV = B \quad (7)$$

Where ADV is the set of all possible advisor agents in the marketplace.

4.4.1 Initialization

By employing agent seniority based on agent lifetime and transaction, novice agents are able to build an informed initial advisor list and therefore make good selections of agents with whom they will engage in transactions, even with no or limited experience of their own.

In this model, each buyer agent b_i maintains a list of advisor agents: A_{b_i}. Upon entering the marketplace, the buyer agent initializes this list by evaluating the seniority of possible advisors and selecting the N agents with the highest seniority. Specifically, the procedure to initialize the list is:

1. Once it enters the marketplace, agent b_i determines the set of all potential advisors. The way in which these candidate advisors are discovered and identified depends on the structure of the marketplace, as well as on the agent's preference. For example, some marketplaces may initially limit an agent's view of the system to a certain set of peer nodes. In others, where agents have a broader view, a particular agent may wish to only consider potential advisors within a certain distance of itself. In some cases, a buyer agent may be able to discover and consider all

advisors in the system. For this discussion, the set of candidate advisors to consider, narrowed from all of the possible advisors in the marketplace is the set *C*:

$$C \subseteq ADV \tag{8}$$

2. For each candidate advisor c_i in the set C, agent b_i calculates the seniority $s_{c_i}(t)$ at the current time *t* as defined by equation (2).
3. The agent selects the *N* candidate advisors with the highest $s_{c_i}(t)$ values as its advisor list A_{b_i}.

$$A_{b_i} \subseteq C \subseteq ADV \tag{9}$$

$$A_{b_i} = \{a_1, a_2, a_3, \ldots a_N\} \tag{10}$$

Where A_{b_i} is the internal advisor list for buyer agent b_i selected from the set of candidate advisors *C* and *N* is the preferred advisor list length.

Once established, the advisor list is maintained for use during advice solicitation.

4.4.2 Internal Trust

In this model, buyer agents make use of both internal trust and reputation information to make trust decisions. To this end, each buyer agent b_i maintains a trust rating for each vendor agent v_j with which it has engaged in a transaction in the past. Trust values are in the continuous range from -1 to 1 where -1 denotes extremely untrustworthy, 1 means extremely trustworthy and 0 indicates a neutral rating.

$$IT_{b_i}^{v_j} \in \mathbb{R}; \quad -1 \leq IT_{b_i}^{v_j} \leq 1 \tag{11}$$

Where $IT_{b_i}^{v_j}$ is the internal trust rating that agent b_i holds for agent v_j.

4.4.3 Advice Solicitation

Once an agent has established its list of advisors and is faced with the decision to buy from a particular vendor agent v_i, it solicits advice in the form of a reputation ranking from each of its advisors. That is, for each advisor a_k in its advisor list A_{b_i}, agent b_i requests advice about the vendor in question, v_j. In this model, it is assumed that only a single value can be expected from any given advisor, since in an open multi-agent-system, agents can be free to employ their own trust management system and may not be able to provide information beyond a simple aggregate rating.

If the advisor has a recommendation about v_j, it provides a rating in the continuous range [-1, 1], where -1 is a very poor rating, 0 is neutral and 1 is a very high rating. More formally:

$$AR_{a_k}^{v_j} \in \mathbb{R}; \quad -1 \leq AR_{a_k}^{v_j} \leq 1 \tag{12}$$

Where $AR_{a_k}^{v_j}$ is the advice rating provided by advisor agent a_k regarding vendor agent v_j.

It is possible that not all advisor agents in the advisor list will respond to the request for advice. An advisor's lack of response may be intentional, either because it does not have prior experience with the vendor in question or because the agent is being deceitful and is hiding information. In other cases, the lack of response may be unintentional, as in the case of network failure, for example. For the sake of clarity, advisor agents who do successfully return a response to the buyer agent are referred to as *witnesses*. As such, the set of witness agents is defined as follows:

$$W_{b_i}^{v_j} = \{a \in A_{b_i} \mid AR_a^{v_j} \; exists\} \tag{13}$$

Where $W_{b_i}^{v_j}$ is the set of witness agents who have provided agent b_i with a rating about vendor v_j; $AR_a^{v_j}$ is the rating provided by agent a and A_{b_i} is the complete list of advisors maintained by buyer b_i.

4.4.4 Rating Aggregation

In order to compute a single trust value, the ratings obtained from the advisors must first be combined to obtain a total reputation rating. As discussed in section 4.2, seniority is an indicator of the degree of influence that a member of the community has within the community. Therefore, advice from agents who have greater seniority should have more weight in the aggregated rating than agents who have less seniority. In the seniority reputation model, this effect is achieved by weighting the rating provided by each advisor with that advisor's contribution to the total seniority of the advisor group.

Additionally, since seniority is a measure of the reliability of an individual within the community, the seniority of the vendor is used as a multiplying factor for the combined reputation rating. The motivation behind this approach is to reward agents who have a developed a good reputation as a long standing, active community member while penalizing agents who have a long history of under-delivering and being fair to the new vendors to the system. Using the vendor's seniority in this way results in the following effect: if a vendor has a developed a high reputation over a long period of time and with many transactions, then the aggregated reputation will be extremely high. Similarly, if the agent has developed a poor reputation over many transactions and a long period of time, their total reputation will be extremely low. Meanwhile, agents with lower seniority will have much more moderate total ratings. As a consequence, new vendors in the system will stand out from vendors who

have a history of performing poorly and will thus be judged more fairly than if both bad vendors and new vendors were assigned low reputation ratings.

Specifically, considering these factors, the total reputation value at the current time *t* is given by:

$$R_{b_i}^{v_j}(t) = \begin{cases} undefined^{4} & if \ \left|W_{b_i}^{v_j}\right| = 0 \\ 0 & if \ \left|W_{b_i}^{v_j}\right| > 0 \ and \ \sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t) = 0 \\ \dfrac{\sum_{a_k \in W_{b_i}^{v_j}} [s_{a_k}(t) \cdot AR_{a_k}^{v_j}]}{\sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t)} \cdot s_{v_j}(t) & if \ \left|W_{b_i}^{v_j}\right| > 0 \ and \ \sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t) \neq 0 \end{cases} \quad (14)$$

Where:

$R_{b_i}^{v_j}(t)$ is the total reputation perceived at time *t* by buyer agent b_i in vendor agent v_j;

$W_{b_i}^{v_j}$ is the set of witnesses, which are advisor agents who provided ratings to b_i about v_j;

$s_{a_k}(t)$ is the seniority of advisor a_k at time *t*;

$AR_{a_k}^{v_j}$ is the rating provided by advisor a_k regarding vendor v_j; and

$s_{v_j}(t)$ is the seniority of vendor v_j at time *t*.

Note that equation (14) includes consideration for two special cases. In the first case, if no witness information is provided and thus the set $W_{b_i}^{v_j}$ is empty, the total reputation rating $R_{b_i}^{v_j}(t)$ is set to be *undefined*. This case is may occur, for example, if the vendor to be rated is new to the system or if there is some sort of network failure that breaks the communication paths between the buyer agent and its advisor agents. In the second case, the cumulative seniority of the advisor agents is 0. This case is extremely unlikely because all of the selected advisor agents would have to have not participated in any transactions in the

[4] The term *undefined* is used here to mean that the value cannot be computed, and therefore no value is assigned. In a software implementation, *null* could be used in lieu of *undefined.*

system. This may occur only at the very start of the e-commerce system. Still, the case is included to maintain the mathematical soundness of the model.

4.4.5 Total Trust Computation

The total trust value, which is used by buyer agents to ultimately determine whether or not a vendor agent is trustworthy, comprises two elements: the internal trust as presented in equation (11) and the total reputation rating as given by equation (14). The confidence that a buyer agent has in each of these components is a personal preference and will vary from agent to agent. In fact, for the same agent, the relative importance of internal trust and reputation rating may fluctuate over time based on a set of criteria, such as the amount of direct experience available or even the agent's own seniority ranking (for further discussion, see section 7.1: Future Work). Therefore, the weighting factor α is introduced to indicate the relative importance of the public reputation to the internal trust rating. It can be set to any value in the continuous range [0,1] depending on the agent's own preference.

In some cases, for example in the rating of a new vendor, neither reputation nor internal trust information is available. In these situations, the total trust value cannot be computed and is therefore deemed undefined. Similarly, on some rare occasions reputation information will be available and not internal trust or vice-versa. In these situations, total trust is set simply to the value that is available.

Formally, these conditions and method for combining the total reputation rating with internal trust to compute a total trust value are expressed as follows:

$$T_{b_i}^{v_j}(t) = \begin{cases} undefined & if\ R_{b_i}^{v_j}(t) and\ IT_{b_i}^{v_j}\ are\ undefined, \\ IT_{b_i}^{v_j} & if\ R_{b_i}^{v_j}(t)\ is\ undefined, \\ R_{b_i}^{v_j}(t) & if\ IT_{b_i}^{v_j}\ is\ undefined, \\ \alpha \cdot R_{b_i}^{v_j}(t) + (1-\alpha)\, IT_{b_i}^{v_j} & otherwise \end{cases} \quad (15)$$

Where:

$T_{b_i}^{v_j}(t)$ represents the total trust that buyer agent b_i has in vendor agent v_j at time t;

α is the confidence that agent b_i has in public reputation versus internal trust;

$R_{b_i}^{v_j}(t)$ is the total reputation rating of vendor v_j as perceived by buyer b_i at time t, following equation (14); and

$IT_{b_i}^{v_j}$ is the internal trust held by buyer b_i in vendor v_j, as in equation (11).

4.4.6 Vendor Selection

When a buyer agent intends to enter into a transaction with a vendor, it must determine which of all the possible vendors is the most trustworthy. Thus, the first step in selecting a vendor is to compute a total trust value for each possible vendor, following (15). Given these values buyer b_i can proceed with identifying three sets of vendors: trusted vendors, untrusted vendors and unrated vendors. This is achieved by comparing each of the total trust values with an agent-defined trust threshold θ.

The trust threshold indicates a value in the continuous range [0,1] that any given trust rating must exceed in order to categorize the associated vendor as a trusted vendor. This approach is incorporated into the proposed seniority trust model to reflect the real-life condition that individuals' risk tolerances vary. In fact, even a single individual's own risk tolerance may vary depending on the stakes of the transaction at hand. By using a simple threshold, it is straight-forward to have an agent set the threshold to a value that best suits their needs given their propensity for risk acceptance and the circumstances of the transaction.

And so, buyer agent b_i reviews all of its total trust values and categorizes potential vendors as follow:

$$V_{trusted} = \{v \in V | T_{b_i}^{v}(t) > \theta\} \quad (16)$$

$$V_{untrusted} = \{v \in V | T_{b_i}^{v}(t) \leq \theta\} \quad (17)$$

$$V_{unrated} = \{v \in V | T_{b_i}^{v}(t) \text{ is undefined}\} \quad (18)$$

Where:

V is the set of all vendors in the marketplace;

$V_{trusted}$ is the set of vendors whose trust ratings exceed the trust threshold;

$V_{untrusted}$ is the set of vendors whose trust ratings are lower than or equal to the trust threshold;

$V_{unrated}$ is the set of vendors for whom trust information could not be computed;

v is the current vendor; and

$T_{b_i}^{v}$ is the total trust that buyer agent b_i has in agent v.

Given these sets of vendors, buyer agent b_i proceeds with identifying the vendor it trusts most for the current transaction, which is referred to as $v_{selected}$. If the list of trusted vendors $V_{trusted}$ is empty, then the buyer agent randomly selects a vendor from the unrated set $V_{unrated}$. On the other hand, if $V_{trusted}$ is not empty, then buyer agent b_i chooses the vendor in $V_{trusted}$ with the highest total trust $T_{b_i}^{v}$ value as its selected vendor $v_{selected}$.

When the buyer agent selects a vendor from the $V_{trusted}$ set, it must capture all of the advice that it had obtained regarding $v_{selected}$ so that this information can be used to perform its post-transaction assessment of its advisors' reliability. This information is stored as a set $\gamma_{v_{selected}}$ of three-tuples, each of which includes an identification number for the advisor agent, the seniority of the advisor agent at the time the recommendation was made, and the corresponding rating from the advisor. That is:

$$\gamma_{v_{selected}} = \{(id(a), s_a(t_a), AR_a^{v_{selected}}) \mid a \in W_{b_i}^{v_{selected}}\} \quad (19)$$

Where:

b_i is the buyer agent who has made the trust decision;

$V_{selected}$ is the vendor agent that b_i has chosen as the supplier for a transaction;

a is an advisor agent in the set of witnesses $W_{b_i}^{v_{selected}}$;

$W_{b_i}^{v_{selected}}$ is the set of witnesses, as defined by equation (13), who are advisors that have provided b_i with a rating for $v_{selected}$;

$id(a)$ is a unique agent identifier number for agent a;

t_a is the time at which agent a provided agent b_i with a rating about $v_{selected}$;

$s_a(t_a)$ is the seniority of the advisor a at the time it provided the rating about $v_{selected}$; and

$AR_a^{v_{selected}}$ is the rating provided by advisor agent a regarding vendor $v_{selected}$.

With the vendor selected and the advice information about the selected vendor stored for future reference, the buyer agent proceeds with the transaction.

4.4.7 Feedback

Once the buyer has engaged in a transaction with the selected vendor, it must use the outcome of the transaction as feedback into its trust system in order to continue to improve results over time. This feedback takes two forms: (i) the utility gained from the transaction must be used to update the internal trust rating for the vendor agent $v_{selected}$; and (ii) the reliability of the advisor agents who provided ratings regarding $v_{selected}$ must be assessed based on the observed results of the transaction. Periodically, the information provided through the feedback process is used to review the buyer agent's advisor list and identify advisors that have proven to be unreliable so that they can be replaced with other advisors who are deemed more credible.

The following section details each of these components of the model's feedback loop.

4.4.7.1 Updating Internal Trust

Based on its most recent interaction with vendor $v_{selected}$, buyer b_i must update its internal trust records. Depending on the agent's preference, this step can be performed either by computing a new single trust value based on the outcome of this transaction and the past internal trust value or by storing a record of the transaction so that the details can be accessed in the future.

The seniority trust model is flexible enough to accommodate either approach. However, for demonstrative purposes, it is suggested that the latter approach be adopted and implemented as follows: the outcome of the transaction is assessed and a satisfaction rating, $SR_{b_i}^{v_{selected}}$, in the continuous range [-1,1], is determined then stored in the buyer agent's internal trust rating table. This satisfaction rating is expressed as follows:

$$SR_{b_i}^{v_{selected}} \in \mathbb{R}\,;\, -1 \leq SR_{b_i}^{v_{selected}} \leq 1 \qquad (20)$$

Using this approach allows the agent to employ its preferred method of combining historical information to compute internal trust ratings. It even allows the agent to modify its internal trust approach over time and further refine its implementation to meet its individual goals.

4.4.7.2 Updating Internal Advisor Rating Table

In addition to updating its internal trust about the vendor selected for a transaction, the buyer agent uses its satisfaction with the results of the transaction to rate the credibility of its advisors. Following [7], this model does not use these credibility scores to weight agents' ratings in the aggregation step. Instead, they are used in the process described in section 4.4.7.3 to determine whether the advisor should remain on the buyer's advisor list or should instead be replaced by a new advisor.

To evaluate the credibility of its advisors, the buyer agent compares the advice received from its advisors to its satisfaction with the transaction outcome. More specifically, buyer agent b_i iterates over each entry in the advice table $\gamma_{v_{selected}}$ defined in equation (19). For each record, it computes a credibility rating $CR_{b_i}^{a_k}$ which is a value in the continuous range [-1,1]. It denotes the credibility that b_i attributes to advisor agent a_k based on the results of b_i's most recent transaction. A credibility rating that approaches 1 indicates a very high degree of credibility and is attained when the advisor's rating $AR_{a_k}^{v_{selected}}$ is very close to the buyer agent's satisfaction rating, $SR_{b_i}^{v_{selected}}$. The credibility rating is calculated as follows:

$$CR_{b_i}^{a_k} = 1 - \left| AR_{a_k}^{v_{selected}} - SR_{b_i}^{v_{selected}} \right| \tag{21}$$

Where:

$CR_{b_i}^{a_k}$ is the credibility rating that buyer agent b_i has toward advisor a_k;

$AR_{a_k}^{v_{selected}}$ is the reputation rating provided by advisor agent a_k regarding vendor $v_{selected}$; and

$SR_{b_i}^{v_{selected}}$ is the satisfaction rating for vendor $v_{selected}$ given by buyer b_i based on their latest transaction.

Each of the credibility ratings computed is stored in the buyer agent's *Advisor Rating Table* λ_{b_i}as part of a three-tuple that also includes an agent identification number for the advisor and the seniority of the advisor at the time that it provided the advice being rated. This table persists to track all advisor credibility ratings over time and is therefore formalized as:

$$\text{Given: } W_{b_i} = \{a \in A_{b_i} \mid a \text{ has provided } b_i \text{ with advice about a selected vendor}\}$$
$$\lambda_{b_i} = \left\{ \left(id(a_k), s_{a_k}(t_{a_k}), CR_{b_i}^{a_k} \right) \;\middle|\; a_k \in W_{b_i} \right\} \tag{22}$$

Where:

W_{b_i} is the set of all advisor agents who have provided advice to b_i about a vendor that b_i has subsequently selected for a transaction;

λ_{b_i} is the Advisor Rating Table for agent b_i;

a_k is the advisor whose credibility rating is included in the current vector;

t_{a_k} is the time at which a_k provided the advice that is being rated;

$id(a_k)$ is the unique identifier number for advisor agent a_k;

$s_{a_k}(t_{a_k})$ is the seniority of a_k at the time that it provided the advice that is being rated; and

$CR_{b_i}^{a_k}$ is the credibility rating of a_k as given by equation (21).

Note that the buyer agent can dispose of its advice table $\gamma_{v_{selected}}$ after this step, since the data it was used to record is no longer required. From this point forward, only the Advisor Rating Table λ_{b_i} is used as a historical measure of advisors' performance.

4.4.7.3 Refreshing Advisor List

From time to time, the buyer agent may wish to refresh its advisor list by removing untrustworthy advisors and replacing them with new advisors. The refreshing period is left up to the agent; however, the detailed steps of the procedure are defined below.

In this process, the seniority metric is used to weight the credibility ratings that have been attributed to a particular advisor agent over time. This application of the seniority metric is significantly different than its use in other parts of the seniority trust model. Here the model considers the seniority of a single agent over time, while in the rating aggregation phase of the model seniority is used to rank the influence of different agents at a single point in time.

Employing the seniority metric in the aggregation of credibility metrics provides two key benefits. First, it correlates the credibility of the advisor to its perceived expertise at the time it was called upon to offer advice. By using seniority as a weighting factor, the model allows a buyer agent to offer a form of forgiveness to ratings obtained when the advisor was somewhat less experienced, yet expect more of the advisor when they have relatively greater experience. Second, given the agent lifetime component of seniority, the seniority of an individual agent always increases over time. Therefore, by weighting credibility ratings with their corresponding seniority values, the most recent credibility ratings will naturally have greater influence on the overall score than will older ratings.

Specifically, the seniority trust model calls for the following steps when a buyer agent refreshes its list of advisors:

1. ***Identify Unreliable Advisors:*** Drawing on the historical information stored in the Advisor Rating Table, the buyer agent first identifies the advisor agents that it deems unreliable. To do so, it must begin by extracting the entries in the Advisor Rating Table λ_{b_i} for each of its advisors; i.e., for each advisor a_k it queries λ_{b_i} for entries that match ($id(a_k)$,—, —) . The set of records for a particular advisor agent a_k is denoted by $\lambda_{b_i}^{a_k}$ and can be expressed as:

$$\lambda_{b_i}^{a_k} = \{(id, s, CR) \in \lambda_{b_i} \mid id = id(a_k),\ a_k \in A_{b_i}\} \tag{23}$$

Where:

$\lambda_{b_i}^{a_k}$ is the set of all credibility scores buyer b_i has available regarding advisor a_k;

b_i is the agent performing the advisor credibility assessment;

a_k is the advisor currently being assessed;

id is the agent identification number stored as part of a three-tuple vector in the Agent Rating Table;

s is the seniority value stored as part of a three-tuple vector in the Agent Rating Table;

CR is the credibility rating stored as part of a three-tuple vector in the Agent Rating Table;

λ_{b_i} is the Agent Rating Table;

id(a_k) is the agent identification number of the advisor agent currently being assessed; and

A_{b_i} is buyer agent b_i's complete list of advisor agents.

From this set of historical credibility ratings for an advisor agent, an overall credibility score is derived. The computation is an average of all credibility ratings in $\lambda_{b_i}^{a_k}$ weighted by the corresponding seniority value, as shown here:

$$CS_{b_i}^{a_k} = \begin{cases} undefined & if \ \left|\lambda_{b_i}^{a_k}\right| = 0, \\ \dfrac{\sum_{(id,s,CR)\in\lambda_{b_i}^{a_k}} (s \cdot CR)}{\sum_{(id,s,CR)\in\lambda_{b_i}} s} & if \ \left|\lambda_{b_i}^{a_k}\right| > 0 \end{cases} \quad (24)$$

Where:

$CS_{b_i}^{a_k}$ is the total credibility score of advisor agent a_k as judged by buyer agent b_i;

b_i is the buyer agent performing the credibility assessment;

a_k is the advisor agent who is the subject of the credibility score;

$\lambda_{b_i}^{a_k}$ is the set of credibility ratings that agent b_i has regarding advisor agent a_k;

s is the seniority of the agent for the current credibility rating; and

CR is the value of the current credibility rating.

Note that the value of $CS_{b_i}^{a_k}$ is undefined when no credibility ratings are available for agent a_k.

Based on the total credibility score for each advisor, the advisors can be sorted into three groups: trusted, untrusted and unrated. Similar to the Vendor Selection process described in section 4.4.6, a threshold value selected by the buyer agent is used to sort the advisors. In this case, the *credibility threshold* is used and is denoted by ω:

$$A_{trusted} = \{a \in A_{b_i} | CS_{b_i}^{a} > \omega\} \quad (25)$$

$$A_{untrusted} = \{a \in A_{b_i} | CS_{b_i}^{a} \leq \omega\} \quad (26)$$

$$A_{unrated} = \{a \in A_{b_i} | CS_{b_i}^{a} \; is \; undefined\} \quad (27)$$

Where:

ω is the credibility threshold,

$A_{trusted}$ is the set of advisors whose credibility exceed the credibility threshold;

$A_{untrusted}$ is the set of advisors whose credibility score is below than or equal to the credibility threshold;

$A_{unrated}$ is the set of advisors whose credibility score is unavailable;

A_{b_i} is the set of advisor agents used by buyer agent b_i; and

$CS_{b_i}^{a}$ is the total credibility score of advisor agent a according to buyer agent b_i.

2. ***Remove Untrusted Advisors:*** The agents in the set $A_{untrusted}$ are, from the buyer agent's viewpoint, are not trustworthy and should be removed from its advisor list. Doing so will protect the buyer agent from receiving unreliable advice from unfavourable advisors in the future. Therefore, the buyer agent reconstructs its advisor list to contain only the agents in the $A_{trusted}$ and $A_{unrated}$ sets.

$$A_{b_i} = A_{trusted} \cup A_{unrated} \quad (28)$$

Where: b_i is the agent refreshing its advisor list; A_{b_i} is the set of advisors for buyer agent b_i; and $A_{trusted}$ and $A_{unrated}$ are the groups of trusted and unrated advisors according to b_i, respectively.

3. ***Blacklist Untrusted Advisors:*** In order to ensure that it never again includes an advisor it knows is not trustworthy to its list of advisors, the buyer agent must blacklist agents who are not untrusted. To do so, buyer agent b_i adds all agents in the set $A_{untrusted}$ to its existing blacklist set:

$$BL'_{b_i} = BL_{b_i} \cup A_{untrusted} \quad (29)$$

Where: b_i is the agent updating its advisor blacklist; BL'_{b_i} is the revised set of advisors which b_i has blacklisted; BL'_{b_i} is the blacklist prior to updating; and $A_{untrusted}$ is the set of unreliable advisors to add to the blacklist.

4. ***Add New Advisors:*** Once the unreliable advisors have been removed from the buyer agent's list of advisors, the list needs to be replenished by identifying new advisor agents and adding them to the list. The first step in this process is to consult with the remaining advisors, if any, for recommendation ratings. The techniques presented in sections 4.4.3 through 4.4.6 are applied. If holes in the advisor list persist even after obtaining recommendations from advisor agents, the buyer agent must seek out other advisors, following the approach outlined for initializing the advisor list in section 4.4.1.

Given this refined set of advisors, the buyer agent is prepared to move forward with applying the seniority trust model to perform more trust and purchasing decisions, constantly improving upon its past results.

Chapter 5

Experimental Evaluation

The effectiveness of the seniority trust model presented in section 4 has been evaluated through software simulation. The objective of this experimental analysis is to determine the validity of using seniority as a metric to enhance trust and reputation systems. Therefore, the seniority metric is tested in isolation of other possible contributing approaches. For comparison purposes, the outcomes of buying agents employing four different test models: *Random Base Model*, *Seniority Base Model*, *Random Model* and *Seniority Model* are examined in four test scenarios: *Balanced Honesty*, *Dishonest Advisors*, *Bad Vendors* and *Casual Buyers*.

This chapter describes the details of the implemented simulation software. The test reputation models are then presented, and finally the experimental results achieved are discussed.

5.1 Simulation Overview

The software simulation used was developed in Java. Sample source code from the software is available in Appendix A.

Following the model presented in section 4.4, the software simulates an electronic marketplace that is populated by buying agents and vendor agents. For the sake of clarity, a single market context is used. In other words, the simulation assumes that all vendors are selling competitive products and that all buyers are procuring a product of that type.

On each time step, every buying agent in the marketplace is given the opportunity to buy, should they chose to do so. Agents may opt out of purchasing if, for example, they are infrequent purchasers. At the end of each time step, the details of the transactions that occurred are stored and the cycle continues.

Each run of the simulation consists of *N_TIMESTEPS* which is defined as one of the simulation parameters. Each scenario, then, is repeated ten times so that the results presented show an average supported by numerous evidence rather than a single anomaly.

5.2 Agent Roles

The simulated marketplace is populated by buyer agents and vendor agents. However, as expressed in equation (7) in the seniority model, buyer agents can also act as advisor agents to other buyer agents. Therefore, there are three distinct and unique roles that an agent can play in the simulation environment: advisor agent, buyer agent and vendor agent.

5.2.1 Advisor Agent

An advisor agent is a buyer agent who, upon request, can provide rating information about another advisor agent or about a vendor agent. Since all buyer agents are advisor agents, in

the software simulation, the BuyingAgent class implements the Advisor interface (as shown in Appendix A).

When an advisor agent object receives a request for a rating about a vendor agent, it returns a value in the continuous range [-1, 1]. The accuracy of the rating depends upon the advisor's assigned honesty level, which can be one of: *honest*, *dishonest_high*, *dishonest_low* or *dishonest_erratic* as summarized in *Table 2*. These advisor types have been selected to emulate real-world advisor tendencies.

An advisor agent of type *honest* simply provides the rating that it would use as its internal trust rating. This models a person whose judgments generally do not deviate from the norm and who is willing to provide an honest account of their past experience with another individual or business.

Table 2: Advisor agent types

Advisor Type	**Reputation Rating Returned**
Honest	Average of internal ratings for seller in question
Dishonest_high	Random value in the continuous range [0, 1]
Dishonest_low	Random value in the continuous range [-1, 0]
Dishonest_erratic	Random value in the continuous range [-1, 1]

An advisor whose honesty level is *dishonest_high* always returns a reputation rating in the range [0,1]. These types of advisor agents simulate people who may not intentionally provide dishonest information, but typically like to be nice in their reviews. They also represent a group of individuals who provide high ratings in order to deceive other agents.

Conversely, advisor agents of type *dishonest_low* return reputation ratings in the continuous range [-1,0] to model individuals who maliciously report negatively about other

agents as well as individuals who simply have a negative disposition and are continuously dissatisfied, regardless of the quality of service provided.

Finally, advisor agents that are designated *dishonest_erratic* provide random feedback in the entire range [-1,1]. These agents are meant to represent malicious agents who provide ratings based on their own personal motivations rather than based on personal experience. Their ratings can be unjustly high or low or they may even, in certain instances, be accurate.

5.2.2 Vendor Agent

Vendor agents are participants in the e-commerce marketplace who intend to sell their product or service to buyer agents. In the software simulation, vendor agents respond to a buyer agent's buy request with single utility value which is a discrete number in the range [0,10]. A very low utility value represents a poor quality of service while a very high value represents a superior transaction outcome. A single utility value is used in this experimental implementation because the seniority trust model to be evaluated is concerned primarily with the selection of advisors and weighting of reputation ratings, rather than the assessment of a vendor from a multi-faceted point of view as done in [18] and [61]. The seniority metric has been devised so that it is complementary to such systems, and could therefore be integrated into a model that measures a transaction's outcome based on multiple attributes, as discussed in section 7.1 Future Work. However, for the purposes of evaluating the contribution of the seniority model only, the simulation software generalizes the quality of service with one utility value.

To determine what quality of service a particular seller will provide, each vendor agent is assigned one of the following types: *good*, *average*, *bad* or *erratic*. Based on this type, for each transaction, the vendor agent returns a utility value that is randomly selected from its

associated utility range as presented in *Table 3*. These types are meant to model the variety of quality of service provided by suppliers in real-world commerce: excellent, mediocre, poor and unpredictable.

Table 3: Vendor agent types

Seller Type	Utility Range
Good	[7,10]
Average	[3,6]
Bad	[0,4]
Erratic	[0,10]

5.2.3 Buyer Agent

The third possible role for an agent in the simulator is as a buyer agent. Buyer agents participate in the e-commerce marketplace in order to make purchases from vendor agents who will deliver a high quality of service. In the software simulation, the objective of buyer agents is to maximize the utility gained by reliably choosing *good* vendor agents.

In real-world e-commerce systems, buyers who engage in transactions wait various periods of time between purchases. As discussed in 4.1.2, in eBay, users' buying frequencies range from over 3,000 transactions per month to only tens of purchases over several years. To model a range of purchasing rhythms, an activity level is assigned to each buyer agent in the simulator. This activity level ranges from *constant* to *very low*, and determines the number of time steps the buyer must wait between purchases, as given in *Table 4*.

Table 4: Buyer agent activity levels

Activity Level	Purchasing Period (time steps)
Constant	0
High	0-2
Medium	3-6
Low	7-9
Very Low	10

Buyer agents within the simulated e-commerce marketplace follow a common algorithm for purchasing, which is depicted in Figure 4. The process is slightly modified for agents who employ either the Random Base Model or the Seniority Base Model, as these do not perform the total trust computation or the advisor revision steps (see Section 5.3 for details about the test models); however, the steps are otherwise identical. The buying algorithm is initiated at every time step of the software simulation, and is executed as follows.

First, the buyer agent must determine whether its advisor list needs to be initialized or refreshed; it will need to be initialized if it does not exist and will need to be refreshed if the preset period for updating the advisor list has passed. If either of these steps is required, the process defined for the agent's particular reputation model is performed, as set out in section 5.3. The agent then decides whether or not to buy based on its assigned buying activity level.

If the agent proceeds with making a purchase, then for each candidate vendor *v*, the buying agent asks all of its advisors for their advice about *v* and then aggregates all of the reputation ratings received. In the Seniority Model and Random Model, it computes a total trust value by combining the total reputation value with an internal trust rating (if it exists) that is based on its own history with vendor *v*.

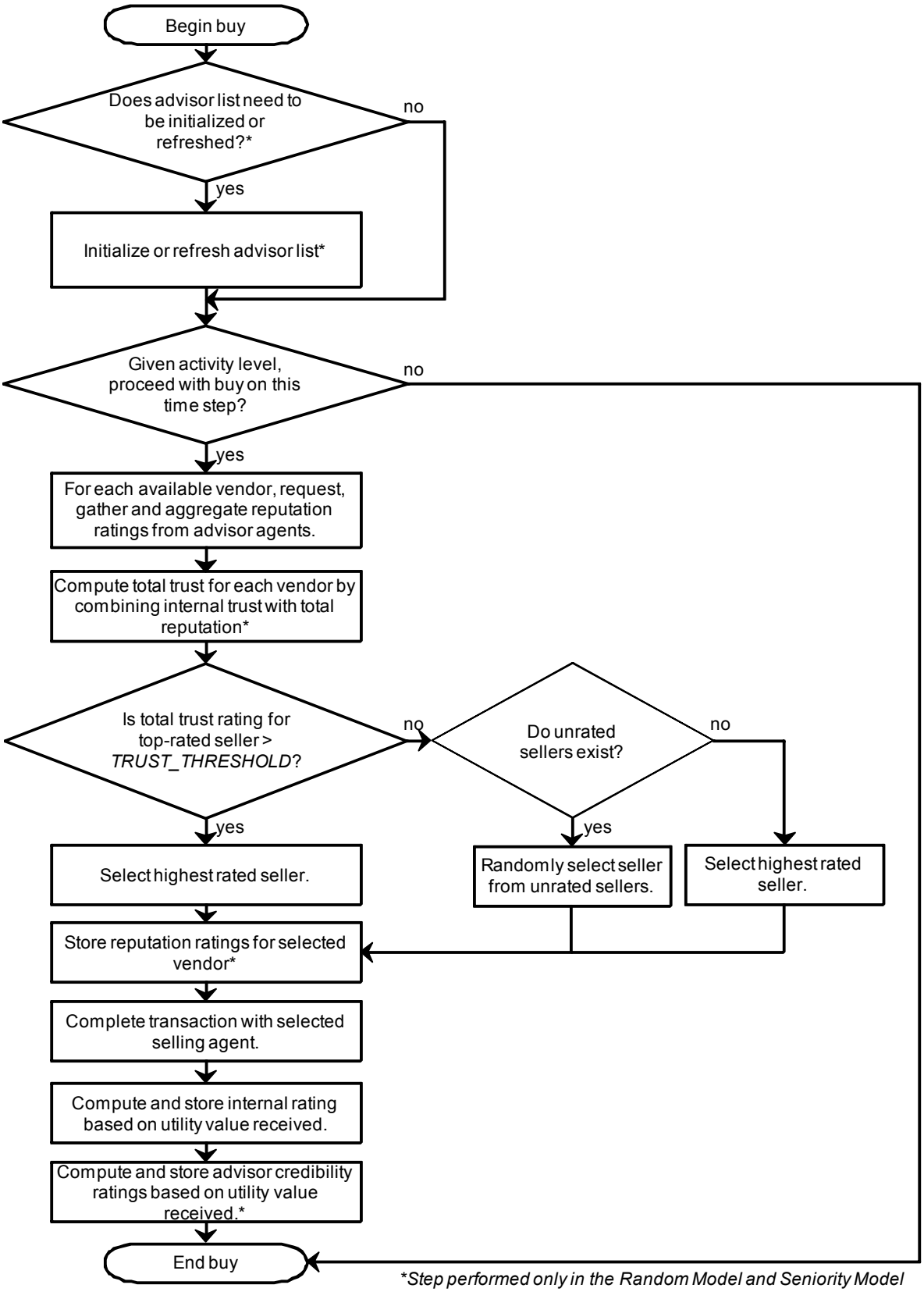

Figure 4: Buying algorithm

With the total trust value of all candidate vendors computed and collected, the buyer agent is prepared to proceed with the selection of a vendor for the current transaction. If the aggregated reputation rating for the most highly rated vendor is greater than the buyer's *trust threshold*, the buyer proceeds with that seller. Otherwise, the agent randomly selects from any unrated sellers or, if necessary, selects the highest rated seller (even though its rating falls below the trust threshold).

The buyer agent then issues a *buy* request to the selected vendor. In response, the buyer agent receives a utility value from the vendor, as described in section 5.2.2. After the transaction is complete, the buyer agent converts this utility value to a trust rating in the continuous range [-1,1] and stores it in its internal rating table for the corresponding vendor.

Finally, in the Seniority Model and the Random Model, the buyer computes and stores a rating for each advisor who had provided a rating for the given vendor. This rating, which records the buyer's satisfaction with the advice received, is in the continuous range [-1, 1] and is persisted in the buyer agent's advisor rating table for use while refreshing the advisor list in the future.

5.3 Test Models

The purpose of the e-commerce simulator is to evaluate the validity of using seniority as a metric in trust and reputation models. In order to test the effect of the seniority metric, it has been isolated as much as possible from other parameters and techniques that might influence the performance of the reputation model. This has been accomplished by testing seniority models that rely solely on seniority for the selection and assessment of advisor agents against models that randomly select advisors and do not perform any weighting techniques. Additionally, to further isolate the seniority metric, the differences between the models are

kept as minimal as possible. In total, four models have been implemented in the simulation software: Random Base Model, Random Model, Seniority Base Model and Seniority Model. For reference, these models are summarized in *Table 5*.

The Random Base Model randomly selects a set of advisors and relies solely on the advice of these agents for trust decisions. Similarly, the Seniority base model selects a fixed set of advisors and uses only their advice for judging the trustworthiness of agents. However, the Seniority Base Model employs the seniority metric in the selection of its advisors and the computation of a total reputation value. By examining the performance of the Seniority Base Model against the Random Base Model, it is possible to determine what contribution, if any, the seniority metric offers to the process of advisor selection.

Table 5: Test models for experimental evaluation

	Random Base Model	**Seniority Base Model**	**Random Model**	**Seniority Model**
Advisor Selection	Random	Seniority	Random	Seniority
Reputation Aggregation	Average	Seniority weighted average	Average	Seniority weighted average with vendor seniority factor
Total Trust Computation	Reputation only	Reputation only	Average of reputation & internal trust	Average of reputation & internal trust
Advisor List Revision • **Credibility assessment**	N/A	N/A	Average of internal advisor credibility ratings	Seniority weighted average of internal credibility ratings
Advisor List Revision • **New advisor selection**	N/A	N/A	Solicitation from advisors and random	Solicitation from advisors and seniority rank

The Random Model and the Seniority Model are designed to investigate the validity of the seniority metric not only for the selection of advisors and reputation aggregation, but also in the evaluation of advisor credibility and advisor list revision. Both of these models follow

completely the buying algorithm presented in Figure 4. However, the Random Model uses random selection of advisors and unweighted averages in its computations, while the Seniority Model makes use of the seniority metric during advisor selection and also in the computation of total reputation, total trust and agent credibility.

In general, these four models have been selected and designed so that the following question can be addressed: In the scenarios of interest, do agents employing the seniority metric perform better than naïve agents who use random advisor selection? If so, the merits of the seniority metric have been demonstrated and further investigation into its use to enhance other models is warranted.

Details on the operation of each of the four test models are presented in the following subsections.

5.3.1 Random Base Model

The Random Base Model is presented to serve as a baseline against which the Seniority Base Model can be compared.

5.3.1.1 Advisor Selection

In the Random Base Model, buyer agents generate their list of advisor agents by randomly selecting *N_ADVISORS* advisor agents from all of the possible advisor agents (i.e., all other buyer agents) in the marketplace.

5.3.1.2 Reputation Aggregation

For the Random Base Model, buyer agents compute the average of all of the ratings received from advisor agents. Therefore, if buying agent *b* is evaluating vendor agent *v*, it solicits advice from all agents in its advisor agent list then computes:

$$R\_RANDOM_{b_i}^{v_j} = \begin{cases} undefined & if \left|W_{b_i}^{v_j}\right| = 0 \\ \dfrac{\sum_{a_k \in W_{b_i}} AR_{a_k}^{v_j}}{\left|W_{b_i}^{v_j}\right|} & if \left|W_{b_i}^{v_j}\right| > 0 \end{cases} \quad (30)$$

Where:

$R\_RANDOM_{b_i}^{v_j}$ is the total reputation perceived by b_i regarding vendor v_j;

$W_{b_i}^{v_j}$ is the set of advisor agents that provided b_i with a rating regarding v_j; and

$AR_{a_k}^{v_j}$ is the rating provided to b_i by advisor agent a_k regarding vendor v_j.

Note that total reputation is undefined when none of agent b_i's advisor agents provide a rating regarding vendor v_j.

5.3.1.3 Total Trust Computation

In the Random Base Model, the only contribution to the trust value attributed to a vendor agent is the reputation information received from advisor agents. Therefore, in the random base case, the total trust value is simply the total reputation computed using equation (30).

5.3.1.4 Advisor List Revision

Advisor list revision is not performed in the Random Base Model; in this case, the same advisors are used throughout an agent's lifetime.

5.3.2 Seniority Base Model

The Seniority Base Model is provided to specifically examine the merit of using the seniority metric in the selection of advisor agents. Its results are meant to be compared with the results obtained via the Random Base Model.

5.3.2.1 Advisor Selection

In this model, when a buyer agent needs to select advisor agents, it first computes the seniority of all candidate advisor agents (i.e., all other buying agents in the marketplace) following equation (2). It then selects the *N_ADVISORS* agents that have the highest seniority values as its list of advisors.

5.3.2.2 Reputation Aggregation

To compute a total reputation rating based on the information provided by its advisor agents, buyer agents using the Seniority Base Model use the seniority of the advisor as a weighting factor. In this way, the computation is similar to that presented in equation (14); however, the Seniority Base Model does not use the vendor's seniority as part of the calculation. Therefore, for the Seniority Base Model, the total reputation is given by:

$$R\_SBM_{b_i}^{v_j}(t) = \begin{cases} undefined & if \ \left|W_{b_i}^{v_j}\right| = 0 \\ 0 & if \ \left|W_{b_i}^{v_j}\right| > 0 \ and \ \sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t) = 0 \\ \dfrac{\sum_{a_k \in W_{b_i}^{v_j}} [s_{a_k}(t) \cdot AR_{a_k}^{v_j}]}{\sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t)} & if \ \left|W_{b_i}^{v_j}\right| > 0 \ and \ \sum_{a_k \in W_{b_i}^{v_j}} s_{a_k}(t) \neq 0 \end{cases} \quad (31)$$

Where:

$R\_SBM_{b_i}^{v_j}(t)$ is the total reputation held at time *t* by buyer b_i in vendor v_j;

$W_{b_i}^{v_j}$ is the set of witnesses, advisor agents who provided ratings to b_i about v_j;

$s_{a_k}(t)$ is the seniority of advisor a_k at time *t*; and

$AR_{a_k}^{v_j}$ is the rating provided by advisor a_k regarding vendor v_j.

As with equation (14), two special cases are considered here: total reputation is deemed undefined when no witness ratings are available and total reputation is 0 when the total seniority of all of the advisors is 0.

5.3.2.3 Total Trust Computation

In the Seniority Base Model, only the total reputation is used to judge trustworthiness. Therefore, the total trust value is equal to the total reputation value given by equation (31).

5.3.2.4 Advisor List Revision

As in the Random Base Model, the buyer agent's advisor list remains unchanged over its lifetime. Thus, advisor list revision is not performed in this model.

5.3.3 Random Model

The Random Model differs from the Random Base Model in its use of internal trust ratings during total trust computation and its inclusion of the advisor list refreshing step. This model is designed to enable the assessment of the effect of using the seniority metric during these two specific steps as well as over the entire buying process.

5.3.3.1 Advisor Selection

Just as in the Random Base Model, in the Random Model buyer agents randomly select *N_ADVISORS* from available advisors in the marketplace.

5.3.3.2 Reputation Aggregation

In this model, a total reputation rating is obtained in the same manner as in the random base case model. Therefore, the total reputation that buyer agent b_i has in vendor agent v_j, $R\_RANDOM_{b_i}^{v_j}$, is computed following equation (30).

5.3.3.3 Total Trust Computation

The total trust that agent b_i has in v_j is determined by combining the total reputation with the corresponding internal trust that b_i has in v_j. It is computed following equation (15), substituting $R\_RANDOM_{b_i}^{v_j}$ for $R_{b_i}^{v_j}$.

5.3.3.4 Advisor List Revision

In the Random Model, advisors are periodically evaluated and potentially replaced in the advisor list based on advisor credibility ratings that are computed after each transaction. The procedure for revising the advisor list is very similar to that described in 4.4.7.3 and can be summarized as follows:

1. The average of all stored advisor ratings is computed;
2. Advisors with an average rating below the *CREDIBILITY_THRESHOLD* are removed from the advisor list and blacklisted so that they will not be added again in the future;
3. If necessary to replenish the list of advisors to *N_ADVISORS*, new advisors are added based on recommendations from other advisors and, if required, through random selection.

5.3.4 Seniority Model

The Seniority Model is the implementation in the software simulation of the model presented in section 4.4. The results of buyer agents implementing this model are meant to be compared against the results of buyers using the Random Model because the two models follow exactly the same buying algorithm. They vary only in the parameters they use in specific computations.

5.3.4.1 Advisor Selection

The Seniority Model follows precisely the steps given in section 4.4.1 to use the seniority metric to select *N_ADVISORS* from amongst all of the other buyer agents in the marketplace.

5.3.4.2 Reputation Aggregation

When a buyer agent employing the Seniority Model receives ratings from advisors, it compiles these into a single total reputation rating by weighting ratings with the corresponding advisor's seniority and by using the vendor's own seniority as a factor. Specifically, the process described in section 4.4.4 is followed and total reputation is therefore computed using equation (14).

5.3.4.3 Total Trust Computation

The total trust value is also computed by following the proposed Seniority Model, and therefore equation (15) is used to combine internal trust with the total reputation. In its computation, 0.5 is used as the reputation factor α, weighting the reputation and internal trust components equally.

5.3.4.4 Advisor List Revision

In the Seniority Model, following section 4.4.7.3, seniority is used to compute a rating for each advisor. If, after removing untrustworthy advisors, more advisors are required to complete the advisor list, the remaining advisors are asked to provide their n (where $n \leq$ *N_ADVISORS*) most highly ranked advisors.

5.4 Simulation Operation

For each run, the simulated marketplace is initialized with a set number of buyers, *N_BUYERS* and vendors *N_VENDORS*. As the buyers are added, the model they implement is determined by iterating over the list of four model types. This ensures that each model has a representation in the simulator that is as equal as possible. Buyer activity level, vendor type and advisor honesty level are assigned randomly as agents are added, while making certain that the proportions specified by the simulation parameters are maintained.

The simulator then executes the time steps of the marketplace, the total number of which is defined by *N_TIMESTEPS*. At each time step, every buyer is given the opportunity to buy, and the results of any resulting transactions are recorded. After every interval *I* of time steps that passes, the simulator adds a new group of buyers to the marketplace. These groups are each assigned a number, herein referred to as a *buyer group,* so that their behavior may be analyzed as a set.

For each experiment, the simulator executes ten runs in succession. Reports of the results obtained for each run as well as a summary of all of the runs are written to disk as spreadsheets for analysis. Samples of these spreadsheets are presented in Appendix B and a graphical representation of the results is given in the following subsection.

5.5 Results

With the software simulator in place, four specific experiments have been devised to analyze the contribution of the seniority metric, particularly in the areas it was designed to address. Therefore, as a whole, the experiments aim to determine what effect, if any, the seniority metric has on: (i) initial advisor selection; (ii) minimizing the experience required to produce successful transaction outcomes; (iii) evaluation of advisor credibility; and (iv)

defense against malicious agents. To do so, the behavior of buyer agents implementing the four test models operating in different scenarios is examined. These scenarios are each investigated in a separate experiment, as follows.

Experiment #1: Balanced Honesty investigates the effect of the seniority metric in an e-commerce system where the proportions of honest to dishonest advisors and good to bad vendors are balanced.

Experiment #2: Dishonest Advisors examines the performance of seniority-based models when faced with an e-commerce environment that is heavily populated by advisors who are dishonest.

Experiment #3: Bad Vendors tests the effectiveness of reputation models based on seniority in an e-commerce marketplace where there is a high proportion of vendors who provide a low quality of service.

Experiment #4: Casual Buyers evaluates the results of models employing the seniority metric when most buyers in the system engage in transactions infrequently, and therefore have minimal experience in the marketplace.

As discussed in section 5.2.3, the buyer agents' objective in the simulator is to obtain maximum possible utility by reliably selecting credible advisors. Therefore, in each of these experiments, the results obtained are presented and examined from four points of view: the average utility gained over all purchases, the average utility gained over buyers' first ten purchases, percentage of all purchases in which good vendors were selected and the percentage of first ten purchases in which good vendors were chosen. By considering the first ten purchases separately, it is possible to observe the effect of the seniority metric on buyer agents that are new to the e-commerce system.

Furthermore, the results are presented based on buyer group. In doing so, it is possible to examine the outcomes as the marketplace matures. When the agents in the first buyer group are added to the marketplace, there are no existing agents. Therefore, all agents in this group are novice and there are no senior agents to call upon for advice. Conversely, when the agents in the tenth buyer group are added, the marketplace is very mature and it is expected that there are many senior agents to adopt as advisors. By examining the results based on buyer group, it is possible to identify the impact that the marketplace maturation has on the success of each model.

The motivation, implementation and results for each of the four experiments are described in the proceeding subsections.

5.5.1 Experiment #1: Balanced Honesty

This scenario created in the first experiment aims to portray a realistic environment, where approximately half of the vendor community provides good quality of service and approximately half of the advisors are honest with their ratings. The specific parameters used in this test case are presented in *Table 6*.

Table 6: Parameters for experiment #1: balanced honesty

Parameter	Value	Parameter	Value
Initial Buyers (*N_BUYERS*)	50	Honest Advisors (%)	50
Vendors (*N_VENDORS*)	100	Dishonest_high Advisors (%)	17
Time steps (*N_TIMESTEPS*)	1000	Dishonest_low Advisors (%)	17
		Dishonest_erratic Advisors (%)	16
New buyer group size	25		
Interval to add buyers (*I*)	100	Buyer Activity Constant (%)	5
		Buyer Activity High (%)	10
Bad Vendors (%)	30	Buyer Activity Medium(%)	70
Good Vendors (%)	30	Buyer Activity Low (%)	10
Average Vendors (%)	30	Buyer Activity Very Low (%)	5
Erratic Vendors (%)	10		
		Advisor List Size (*N_ADVISORS*)	10
Vendor Trust Threshold	0.75	Advisor Refresh Cycle (*J*)	10
Advisor Credibility Threshold	0.75		

The results obtained in this experiment are presented in Figure 5 through Figure 8. From these results, it is clear that in this scenario the Seniority Model yields better results than the other models. The Seniority Base Model provides higher utility than the random base case by selecting good vendors more reliably. Likewise, the Seniority Model outperforms the Random Model, as shown in all four figures.

A single exception occurs in the first ten purchases by the first buying group (see Figure 8). For that single point, the Seniority Base Model produces the poorest outcome of all of the models. This is due to the fact that within the first few time steps of the simulator, when the group 1 buyer agents are being added to the marketplace and the very first transactions between buyers and vendors occur, the seniority rating of all agents is similar and very low.

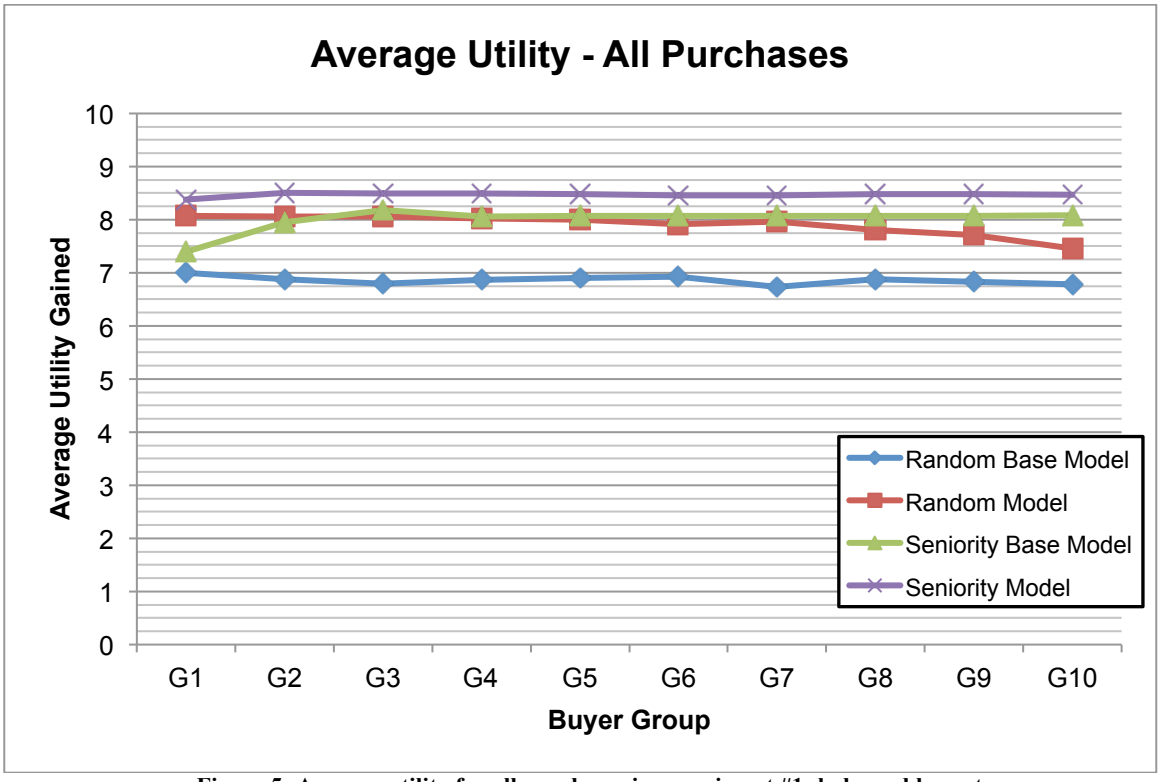

Figure 5: Average utility for all purchases in experiment #1: balanced honesty

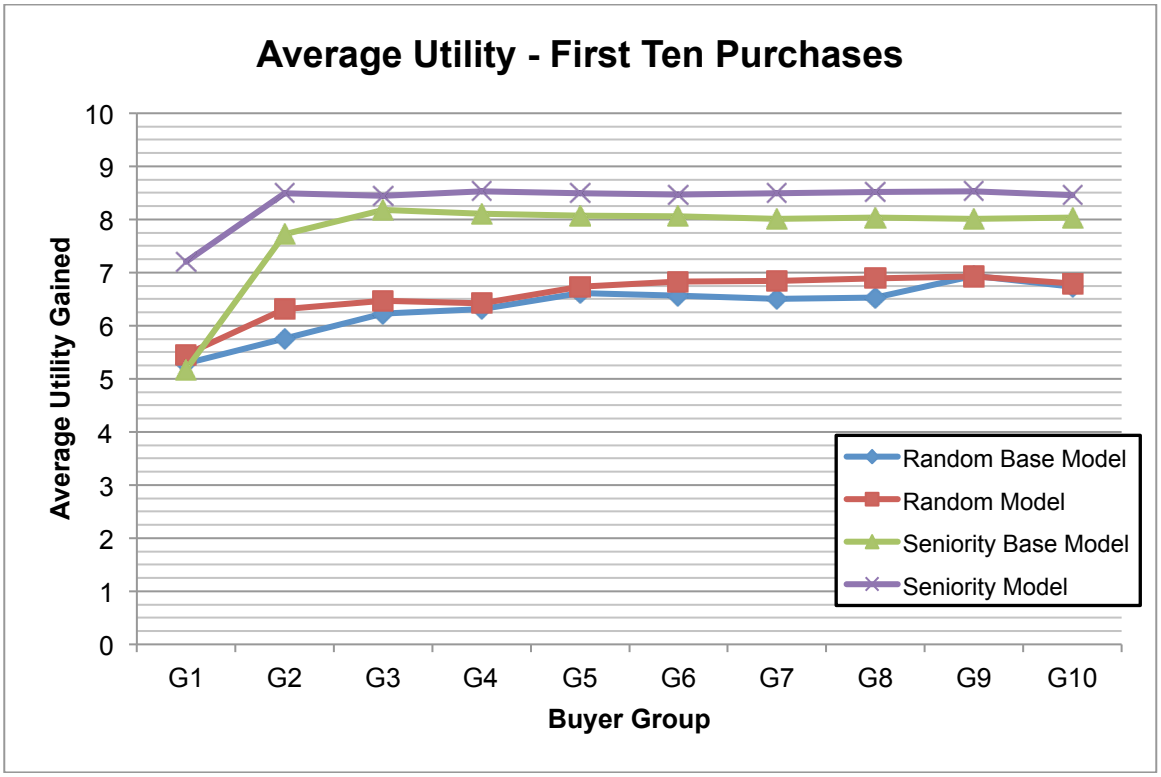

Figure 6: Average utility for first ten purchases in experiment #1: balanced honesty

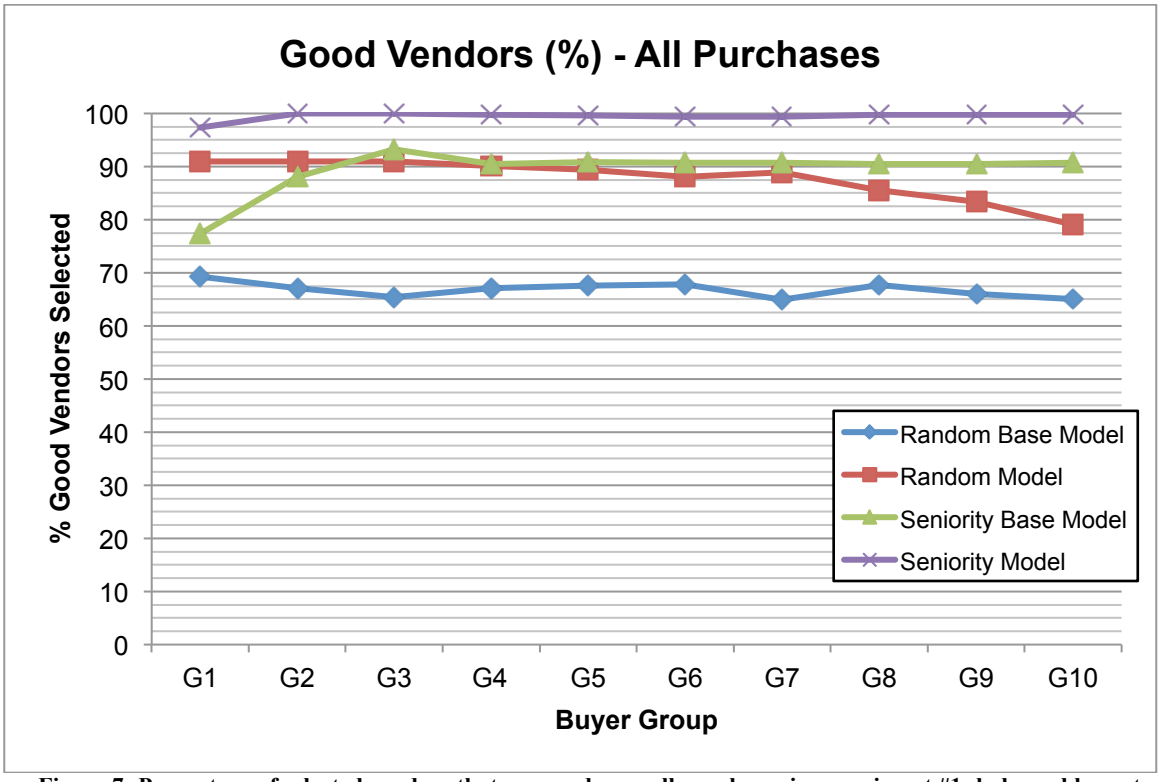

Figure 7: Percentage of selected vendors that are good over all purchases in experiment #1: balanced honesty

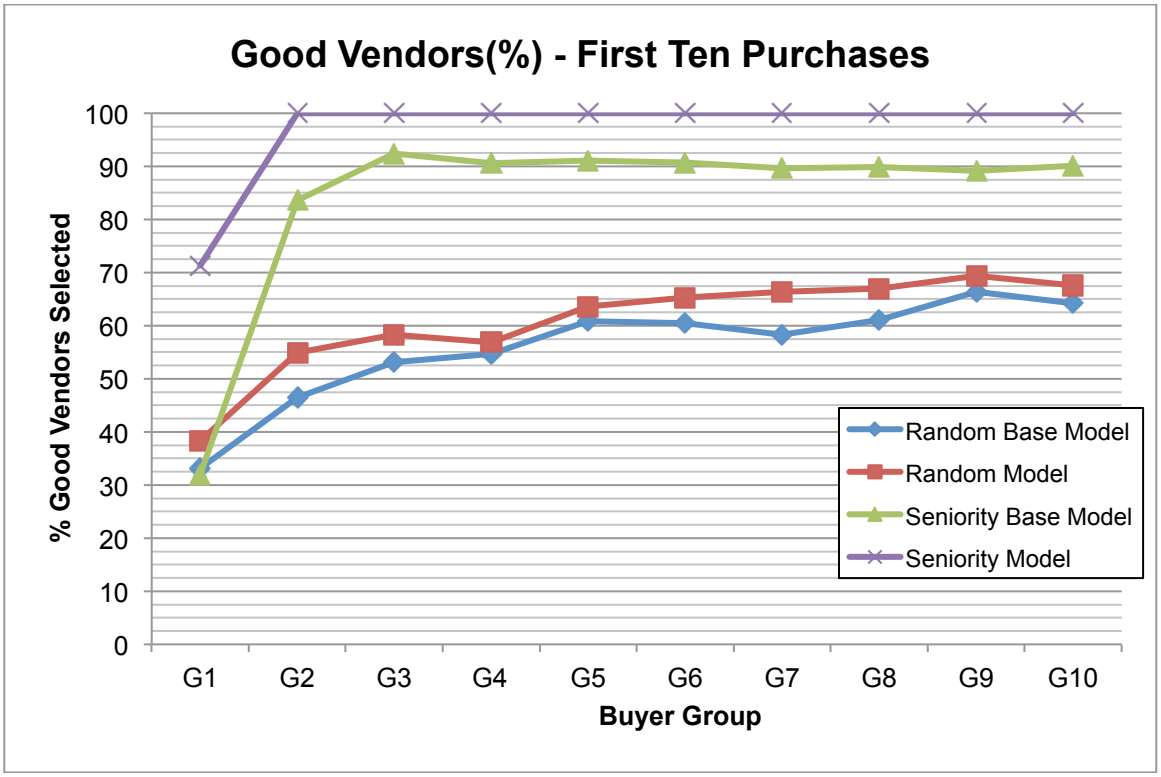

Figure 8: Percentage of selected vendors that are good over first ten purchases in experiment #1: balanced honesty

As a result, the first group of buyers who use the Seniority Base Model cannot reliably depend on seniority ranking and must instead effectively select their advisors randomly from the other buyers.

At that same point, the Seniority Model performs exceptionally well compared with the other models. 71.3% of the vendor agents selected by Seniority Model agents in group 1 were *good* vendors, while the rate was only 31.9%, 33.1%, and 38.3% for the Seniority Base Model, Random Base Model and Random Model, respectively. The possible reasons for this effect lie in the elements of the Seniority Model which are distinct from all of the other models collectively: the use of the vendor's seniority rating in the total reputation computation and the use of seniority in refreshing the advisor list. It is believed that the vendor's seniority rating is the significant contributing factor since the weight of an incremental difference (one transaction, for example) is considerable when the total number of transactions is low. Additionally, even given the relatively low advisor refreshing period of 10 time steps used in this experiment, the number of advisor list revisions that would occur and therefore impact the results of the first ten purchases is low. To validate this explanation and to be certain of the precise factor that resulted the exceptionally high results for the Seniority Model, further experimental analysis is required.

Even beyond the first ten purchases, the Seniority Model yields greater results than the models for all groups, from both the utility gained and good vendor selection points of view. Overall, the Seniority Base Model and the Random Model fare similarly. However, by combining the merits of each model, the use of the seniority metric from the Seniority Base Model, and the use of direct trust and advisor list revision from the Random Model, the results are drastically improved for both novice agents and all agents in the system.

5.5.2 Experiment #2: Dishonest Advisors

The objective of this experiment is to evaluate how well agents employing the Seniority Model perform when the majority of advisors provide wrong or misleading advice. In order to do so, only 25% of all advisor agents in the simulation marketplace are directed to be *honest*. As shown in *Table 7* the remaining 75% are either *dishonest_low*, *dishonest_high* or *dishonest_erratic* (see section 5.2.1 for a detailed description of each type).

Table 7: Parameters for experiment #2: dishonest advisors

Parameter	Value	Parameter	Value
Initial Buyers (*N_BUYERS*)	50	Honest Advisors (%)	25
Vendors (*N_VENDORS*)	100	Dishonest_high Advisors (%)	25
Time steps (*N_TIMESTEPS*)	1000	Dishonest_low Advisors (%)	25
		Dishonest_erratic Advisors (%)	25
New buyer group size	25		
Interval to add buyers (*I*)	100	Buyer Activity Constant (%)	5
		Buyer Activity High (%)	10
Bad Vendors (%)	30	Buyer Activity Medium(%)	70
Good Vendors (%)	30	Buyer Activity Low (%)	10
Average Vendors (%)	30	Buyer Activity Very Low (%)	5
Erratic Vendors (%)	10		
		Advisor List Size (*N_ADVISORS*)	10
Vendor Trust Threshold	0.75	Advisor Refresh Cycle (*J*)	10
Advisor Credibility Threshold	0.75		

The question of how the Seniority Model would fare in an environment where most agents are dishonest is significant. It is a scenario that is conceivable in real-world e-commerce systems and therefore must be adequately accommodated for by the reputation model. However, the seniority method does not employ reputation-based tactics to assess the credibility of the candidate advisor agents before selecting them as agents and to weight the

ratings they provide. Only the agents' age and activity level are applied. Therefore, there is a considerable risk that a dishonest agent could exist in the marketplace for a long time, perform a large number of transactions and then exercise a great deal of influence on other agents in the marketplace.

In Figure 9 through Figure 12 the results collected from this experiment are presented. By examining the results obtained over all purchases—see average utility gained in Figure 9 and percent good vendors selected in Figure 11—it is possible to observe that the models employing seniority produced better results than their random counterparts. That is, the Seniority Base Model performed better than the Random Base Model and the Seniority Model performed better than the Random Model. This indicates that the seniority metric contributes positively to the overall success of agents, even when the marketplace is filled with dishonest advisors.

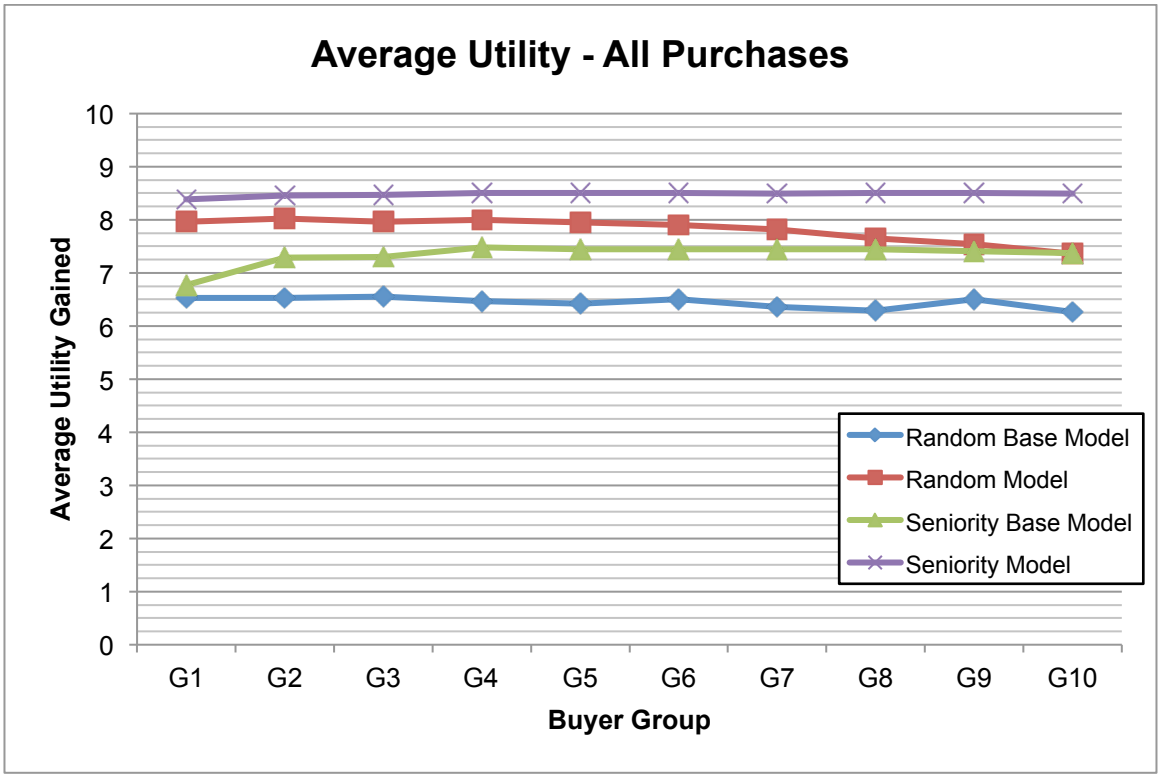

Figure 9:Average utility for all purchases in experiment #2: dishonest advisors

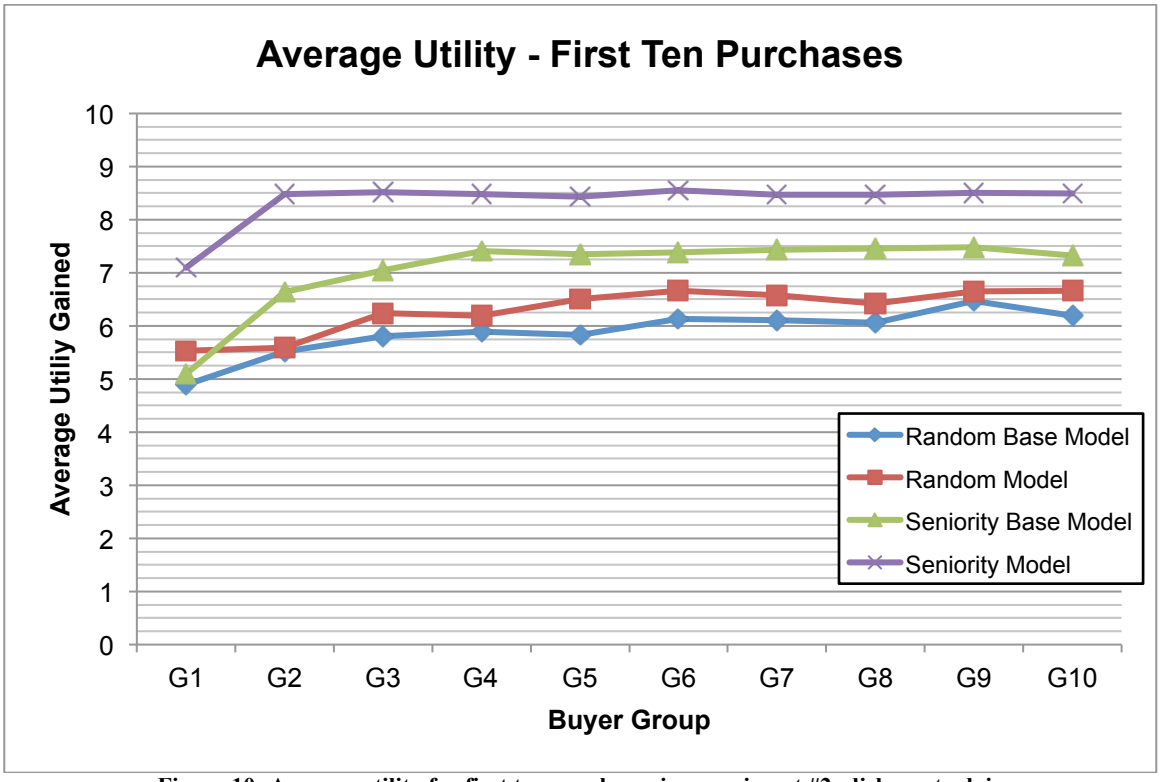

Figure 10: Average utility for first ten purchases in experiment #2: dishonest advisors

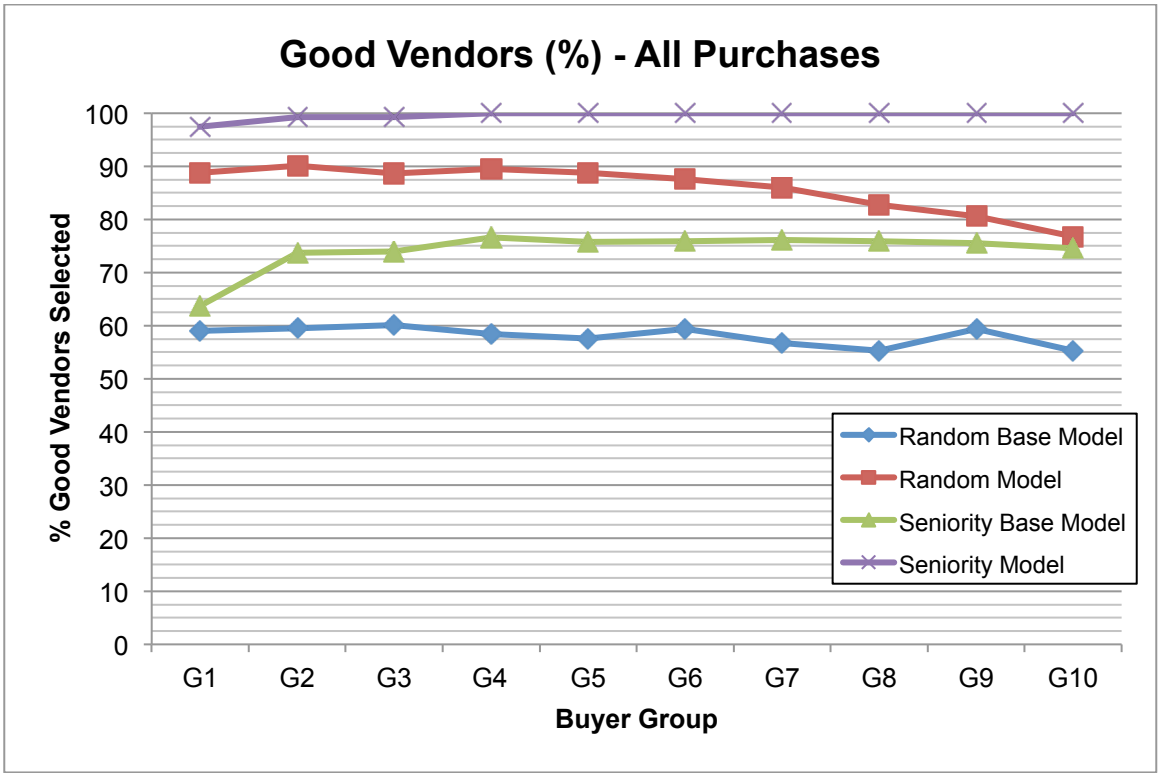

Figure 11: Percentage of selected vendors that are good over all purchases in experiment #2: dishonest advisors

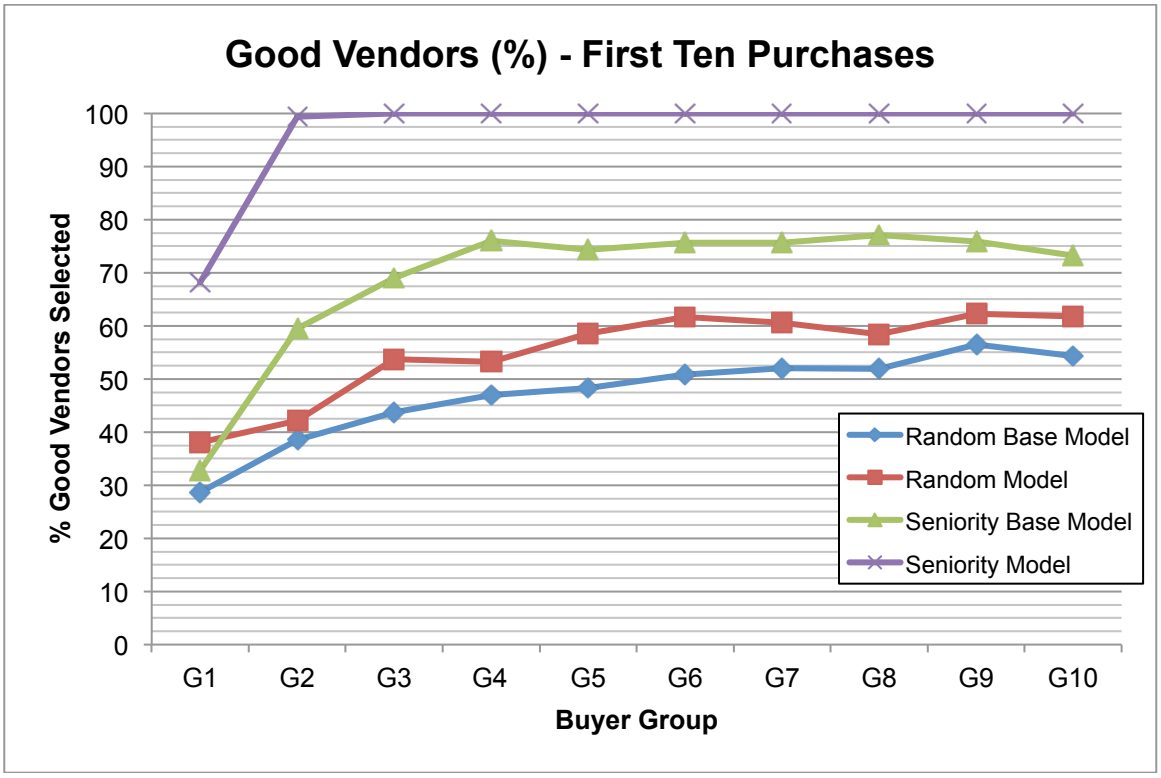

Figure 12: Percentage of selected vendors that are good over all purchases in experiment #2: dishonest advisors

By further examining the results of all purchases in Figure 9 through Figure 12, it is observed that the Seniority Model and Random Model yield better outcomes than the base models. These results indicate the importance of self-reliance, specifically in the form of using internal trust and credibility ratings for advisors, in an environment polluted by dishonest advice.

In reviewing the results obtained over the course of the first ten purchases, as seen in Figure 10 and in Figure 12, the importance of employing seniority becomes apparent. In both views, it is clear that the seniority models outperform the random models. Thus, the seniority metric is shown to be a contributing factor to making improved trust decisions upon entering the marketplace. The seniority metric can also be very effective at deflecting dishonest advice in the early stages of the marketplace, since advisors who have a seniority

rating of 0 (i.e., have not yet performed any transactions) yet provide advice are known to be lying. Following the Seniority Model, these ratings are weighted by the advisor's seniority (zero in this case) and therefore are disregarded.

Overall, the results obtained through this experiment demonstrate that the proposed approach of applying the seniority metric has merit. Even in cases where the majority of advisor agents in the e-commerce system are dishonest, the seniority metric enables agents to identify credible advisors and weight their ratings in such a way that it produces highly successful results.

5.5.3 Experiment #3: Bad Vendors

In the third experiment, the goal is to determine the validity of the seniority metric when most of the vendors in the marketplace provide an unsatisfactory quality of service. To test this scenario, the simulator in this case is initialized with 75% of vendors labeled *bad*.The honesty of the advisors is moderate, with half being honest, as shown in the complete set of parameters provided in *Table 8*.

The objective of any trust and reputation system is to judge with some certainty the character of an agent such that the probability of a positive outcome can be determined. In an e-commerce context, these systems are applied, in part, to assist buyers in identifying reliable suppliers. This task, of course, becomes increasingly difficult as the ratio of bad agents to good agents in the marketplace grows. Therefore, a critical test of any reputation model or any of its components is its performance within an environment that is overrun by bad vendors.

The results from this experiment, as depicted in Figure 13 to Figure 16, show that even in a marketplace dominated by bad sellers, buyer agents employing the Seniority Model

Table 8: Parameters for experiment #3: bad vendors

Parameter	Value	Parameter	Value
Initial Buyers (*N_BUYERS*)	50	Honest Advisors (%)	50
Vendors (*N_VENDORS*)	100	Dishonest_high Advisors (%)	15
Time steps (*N_TIMESTEPS*)	1000	Dishonest_low Advisors (%)	15
		Dishonest_erratic Advisors (%)	20
New buyer group size	25		
Interval to add buyers (*I*)	100	Buyer Activity Constant (%)	5
		Buyer Activity High (%)	10
Bad Vendors (%)	75	Buyer Activity Medium(%)	70
Good Vendors (%)	10	Buyer Activity Low (%)	10
Average Vendors (%)	10	Buyer Activity Very Low (%)	5
Erratic Vendors (%)	5		
		Advisor List Size (*N_ADVISORS*)	10
Vendor Trust Threshold	0.75	Advisor Refresh Cycle (*J*)	10
Advisor Credibility Threshold	0.75		

perform significantly better than agents using the other models. Just as in the previous two experiments, these results indicate that the seniority metric presents value for novice agents, as well as agents who have experience in the e-commerce system. By examining the average utility obtained over all purchases, it can be seen that the Seniority Base Model results are significantly higher than the Random Base Model. Note that the trend for the Seniority Model improves as the average maturity of the agents in the e-commerce marketplace increases. In contrast, the Random Model actually trends downward as the maturity of the system increases. Further research and experimental analysis are required to definitively determine the cause for this diminishing effectiveness in the Random Model.

Supporting the results of the preceding experiments, the outcomes for the first ten

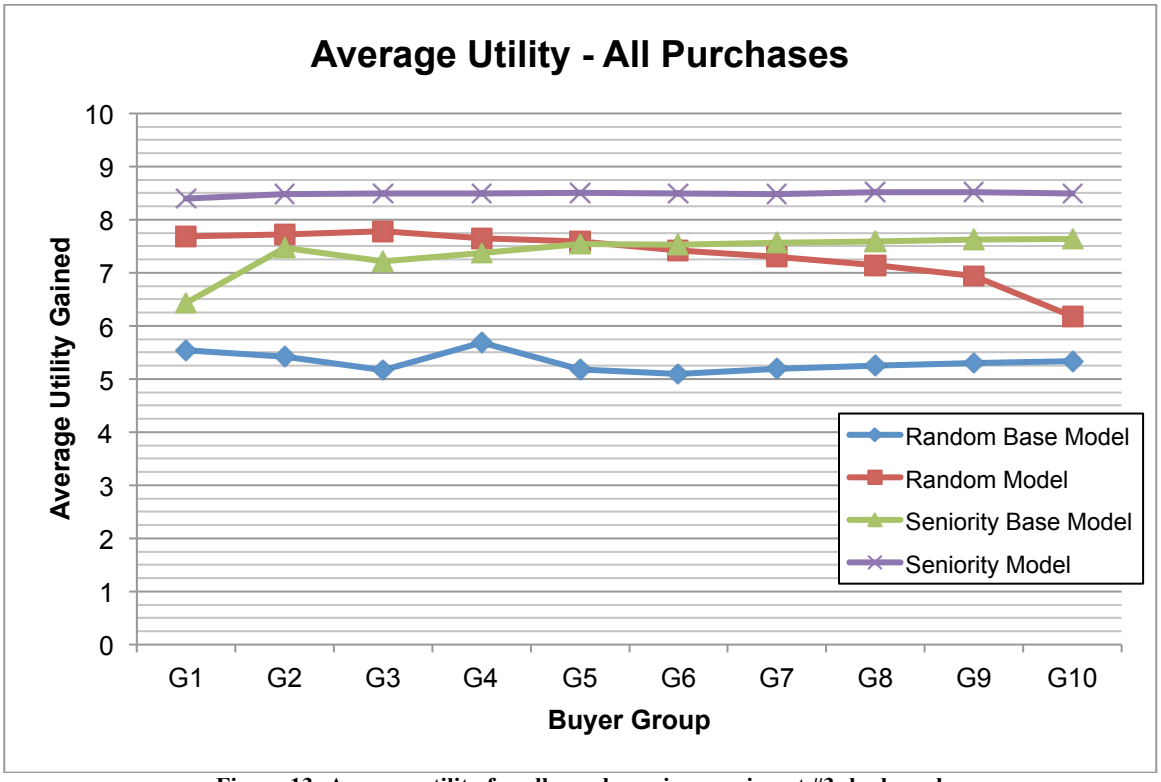

Figure 13: Average utility for all purchases in experiment #3: bad vendors

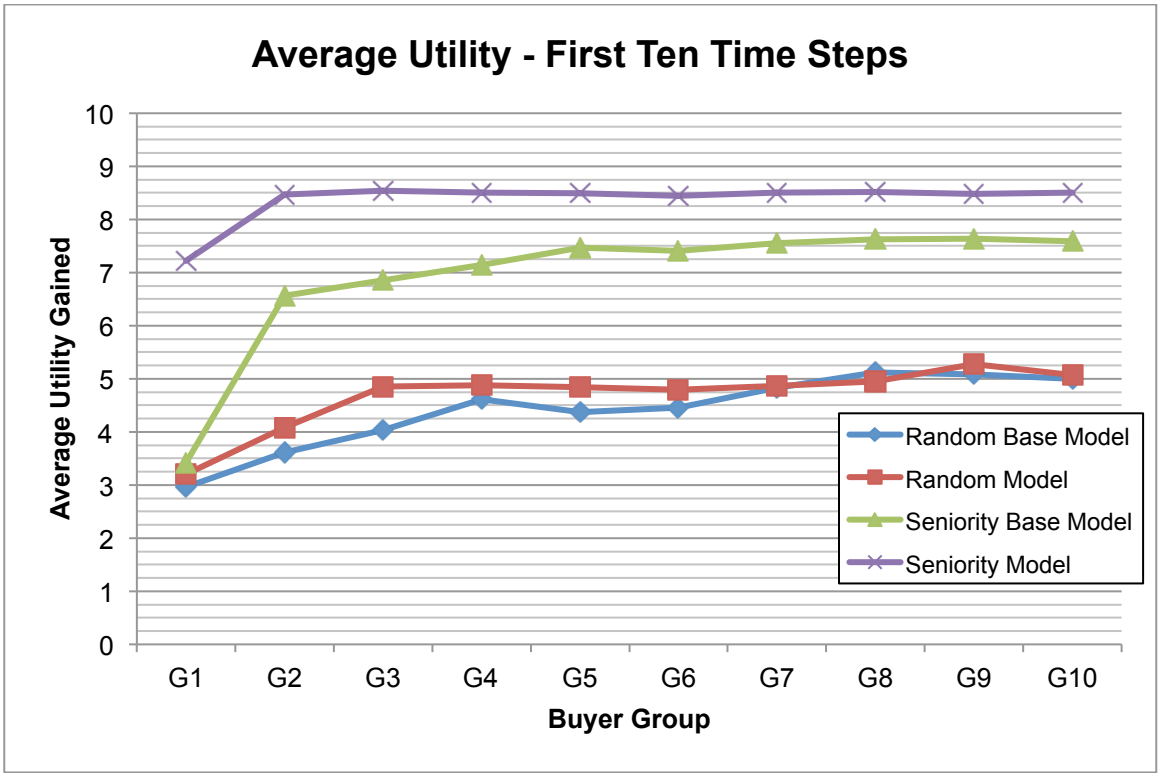

Figure 14: Average utility for first ten purchases in experiment #3: bad vendors

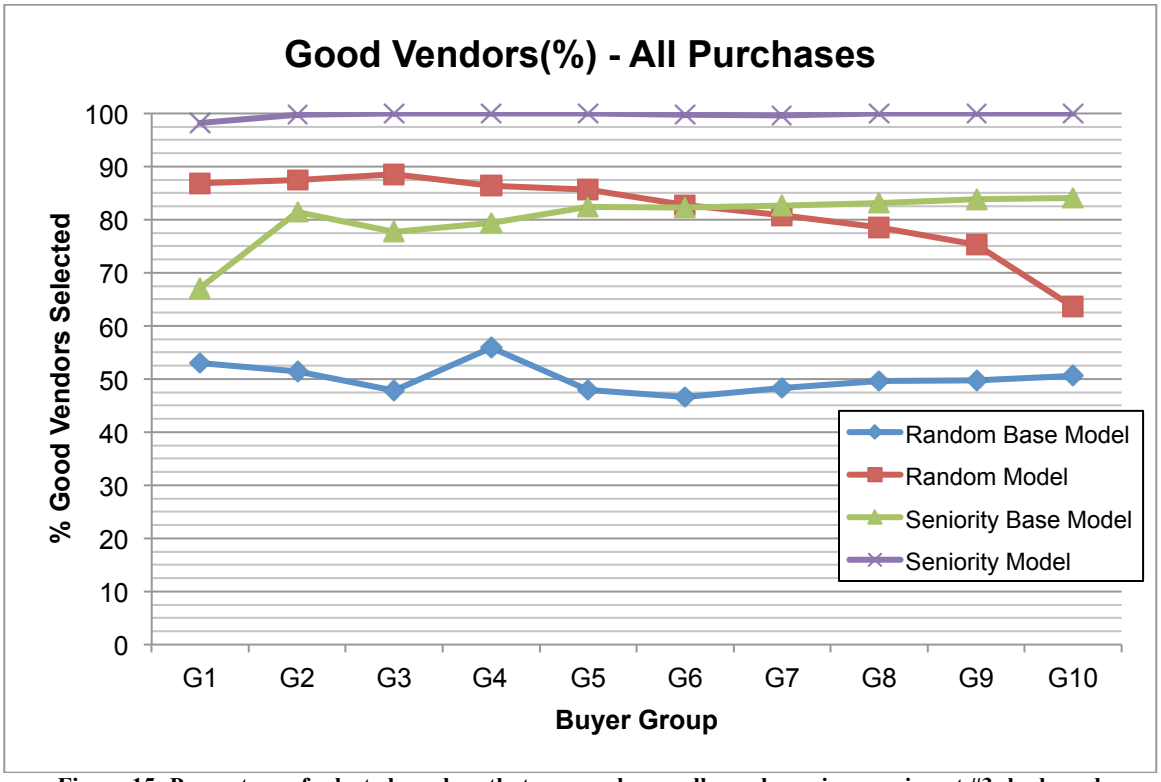

Figure 15: Percentage of selected vendors that are good over all purchases in experiment #3: bad vendors

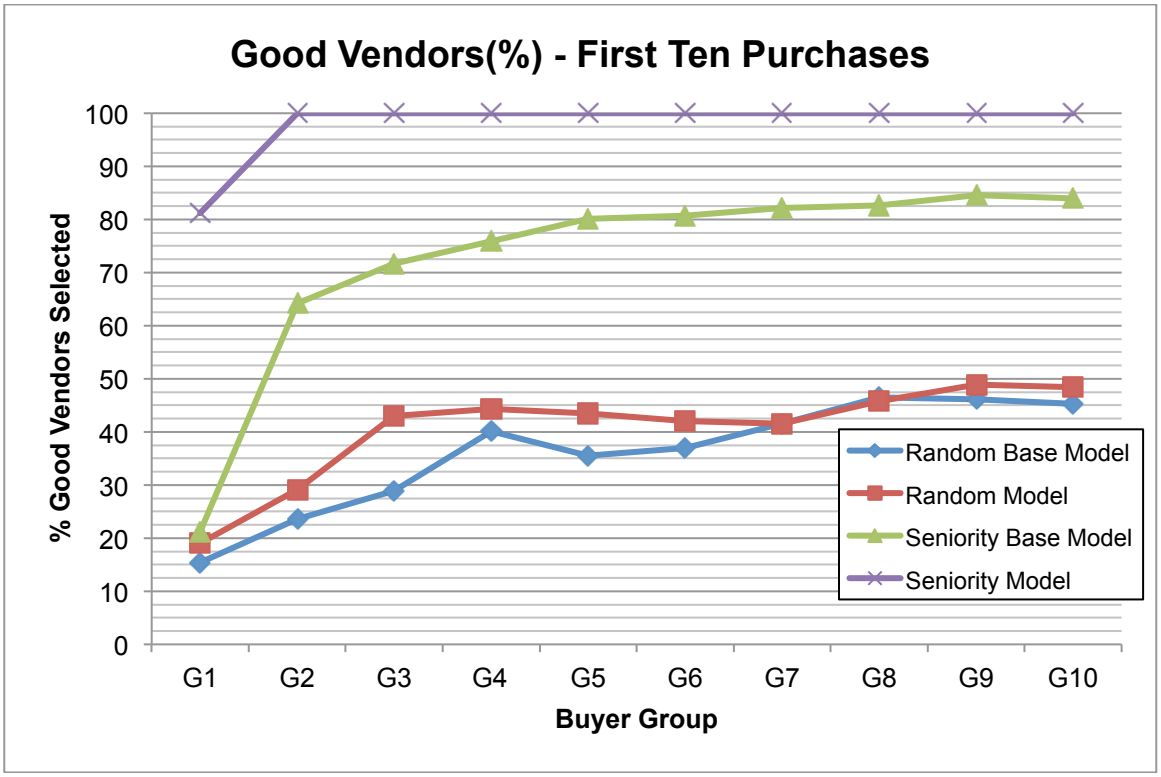

Figure 16: Percentage of selected vendors that are good over all purchases in experiment #3: bad vendors

purchases made, as shown in Figure 14 and Figure 16, indicates that the seniority metric contributes considerably towards novice agents achieving successful results. Even without any prior knowledge of vendor agents or other advisor agents in the marketplace, buyer agents using the models that implement the use of the seniority metric are able to seek out reliable vendors and benefit highly from engaging in transactions with them. Given the results of this experiment, this is true even when bad vendors dominate the e-commerce system in which the buyer agents operate.

5.5.4 Experiment #4: Casual Buyers

The fourth experiment's aim is to determine what contribution, if any, the seniority metric has toward the success of buyer agents that use the e-commerce system infrequently. One of the principle motivations for the inception of the seniority metric is to develop a means of judging advisor credibility that is not dependent upon the buyer agent's past experience. Therefore, the results of this experiment are extremely important in supporting or discounting the validity of the seniority metric.

To model an environment in which buyers exhibit a casual buying frequency, the simulation parameters are adjusted so that the buying activity level of 75% of buyer agents is *Low* and for 5% is *Very Low*. The honesty of advisors and integrity of vendors are balanced, as shown in *Table 9*.

The results presented in Figure 17 through to Figure 20 illustrate the same finding in this scenario as has been observed in the preceding three test cases: the seniority models generate better results for buyers than the equivalent random models. This is especially true in the first ten purchases, where it can be noted that the Seniority Base Model and the Seniority Model yield significantly higher value outcomes than the random models.

Table 9: Parameters for experiment #4: casual users

Parameter	Value	Parameter	Value
Initial Buyers (*N_BUYERS*)	50	Honest Advisors (%)	50
Vendors (*N_VENDORS*)	100	Dishonest_high Advisors (%)	17
Time steps (*N_TIMESTEPS*)	1000	Dishonest_low Advisors (%)	17
		Dishonest_erratic Advisors (%)	16
New buyer group size	25		
Interval to add buyers (*I*)	100	Buyer Activity Constant (%)	5
		Buyer Activity High (%)	5
Bad Vendors (%)	30	Buyer Activity Medium(%)	10
Good Vendors (%)	30	Buyer Activity Low (%)	75
Average Vendors (%)	30	Buyer Activity Very Low (%)	5
Erratic Vendors (%)	10		
		Advisor List Size (*N_ADVISORS*)	10
Vendor Trust Threshold	0.75	Advisor Refresh Cycle (*J*)	10
Advisor Credibility Threshold	0.75		

In this test scenario, it is observed that the Seniority Base Model outperforms the Random Model, even over all purchases. This is in contrast with the results obtained in the other three experiments, where the Seniority Base Model performed as equally well as or even slightly worse than the Random Model, as can be seen by comparing Figure 19 with Figure 7, Figure 11 and Figure 15. In the other experiments, especially in Dishonest Advisors and Bad Vendors, the internal trust and refreshing of the advisor list that are part of the Random Model play a crucial role in defending against the dishonest and bad agents. However, in this experiment the seniority metric used in the Seniority Base Model is the critical element, as it serves as a tool for buyer agents when their own personal experience is lacking. In all of the experiments, the Seniority Model proved to deliver the best results, since it incorporates and enhances the strengths of both the Seniority Base Model and the Random Model.

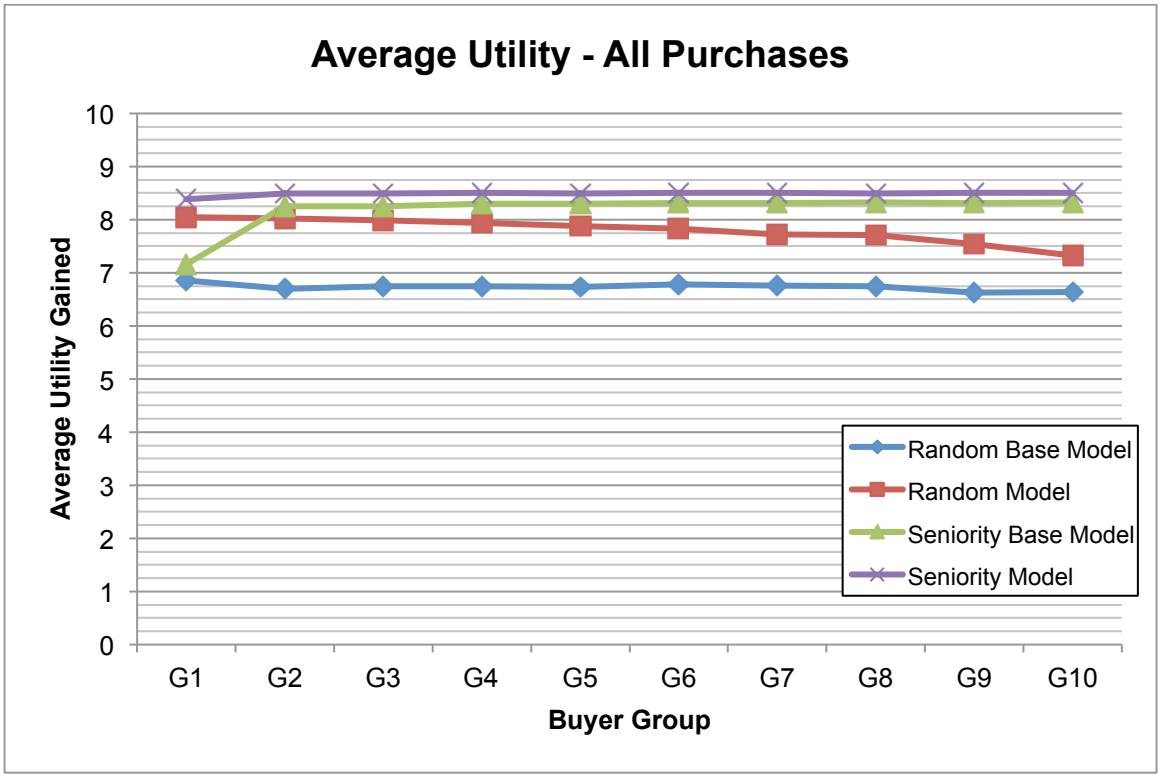

Figure 17: Average utility for all purchases in experiment #4: casual buyers

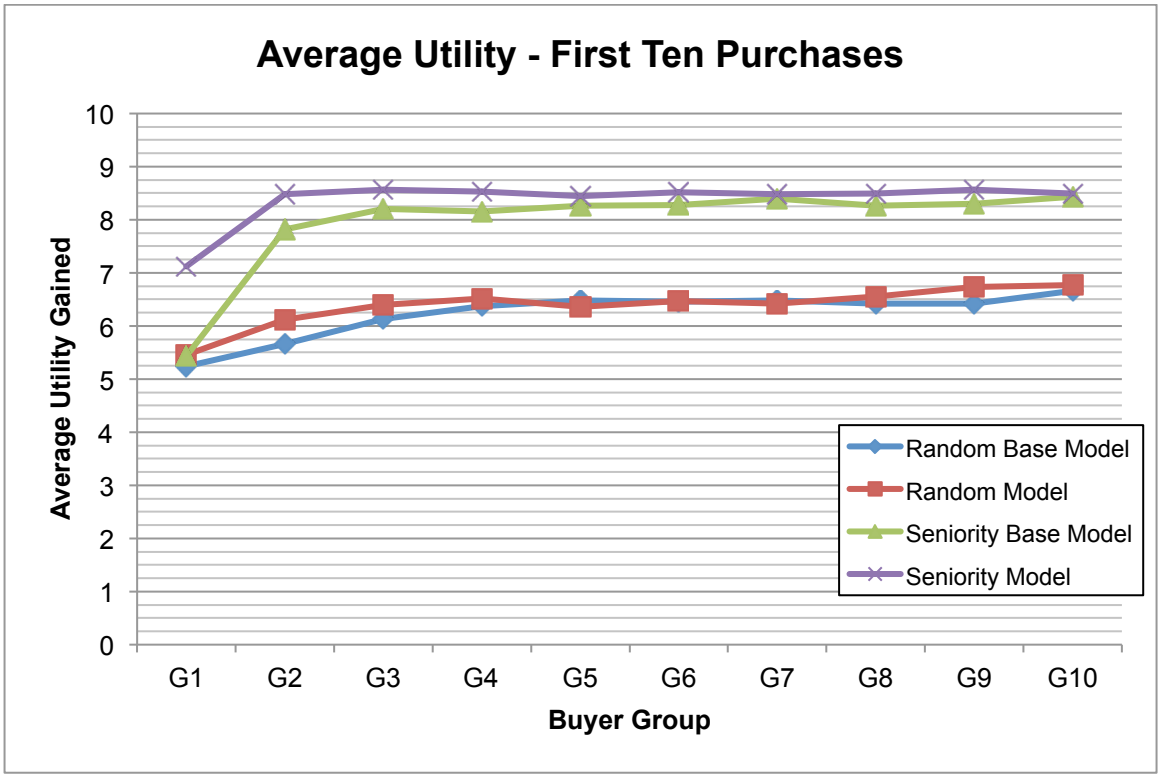

Figure 18: Average utility for first ten purchases in experiment #4: casual buyers

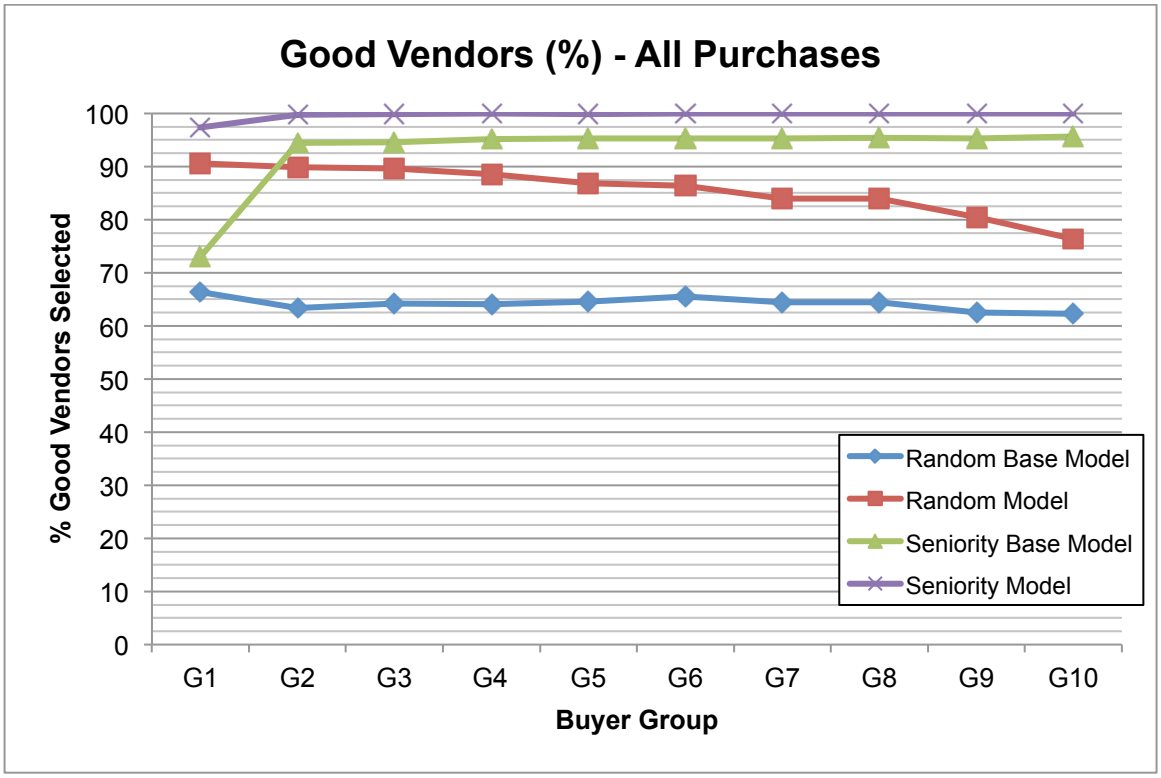

Figure 19: Percentage of selected vendors that are good over all purchases in experiment #4: casual buyers

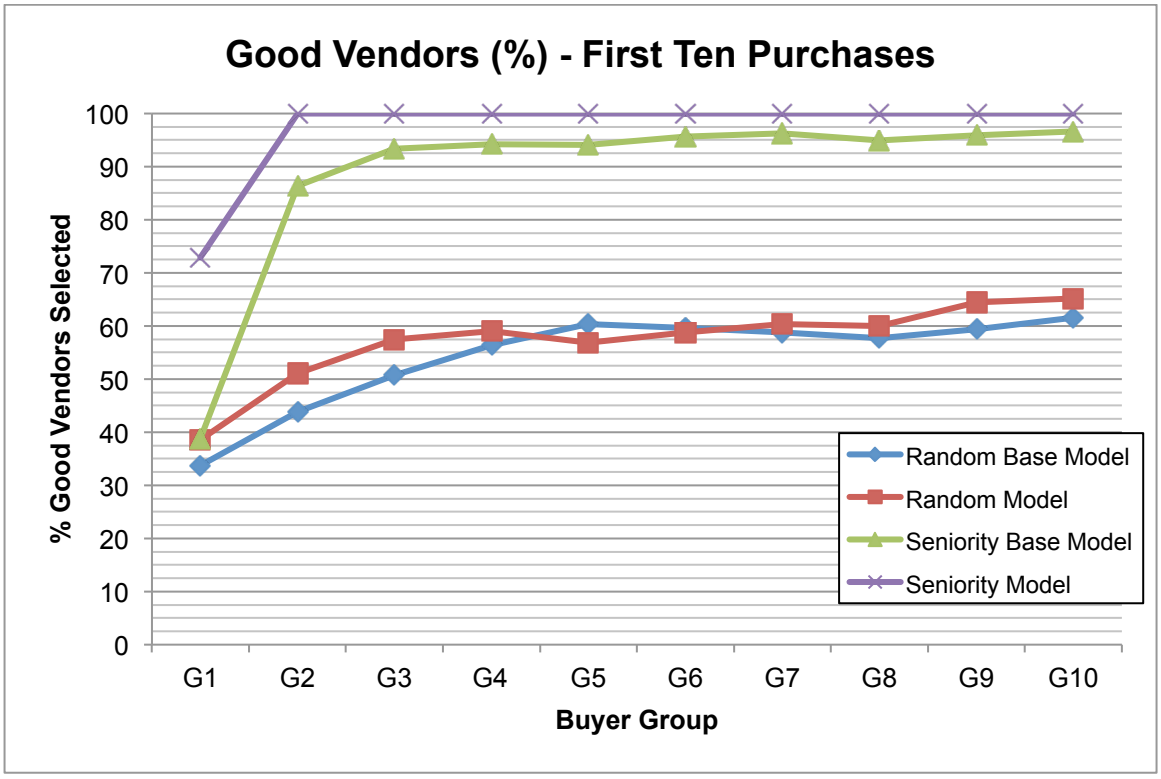

Figure 20: Percentage of selected vendors that are good over first ten purchases in experiment #4: casual buyers

5.5.5 Summary

The results of the experiments clearly indicate the merit of employing seniority as a metric in trust models. In all of the scenarios presented, the Seniority Model produced extremely successful results, exceeding the performance of the other three test models.

By reviewing the results obtained in the first ten purchases in all of the results it is possible to see the significance of utilizing seniority as a new metric in the e-commerce. In all three experiments, the Seniority Base Model and the Seniority Model showed substantial improvement over the random approaches in the first ten purchases. In experiments #1 and #4, where the honesty and integrity of the advisors and bad vendors are balanced, the Seniority Base Model performs almost as well as the Seniority Model in the first ten purchases. However, in experiments #2 and #3, the gap in their results widens. This indicates that a unique application of the seniority metric in the Seniority Model, such as applying the vendor's seniority as a weighting factor, assists in defending against poor behavior on the part of other agents. Upon examination of the outcomes generated over all purchases, the Seniority Model again emerges as the top performer of the four models.

Given all of these results, the following generalizations can be made. (i) The use of the seniority metric to identify credible advisor agents is effective for the short-term gain of new advisors and long-term gains of casual buyers. (ii) The use of vendor's seniority to weight its reputation and of advisor's seniority to revise the advisor list contribute positively to the long-term success of a buyer agent, even when faced with high proportions of unfavorable vendors and advisors. Chapter 6 draws upon these results and continues the discussion on the merits of the seniority metric, particularly as it directly addresses the problems that motivated its inception.

Chapter 6

Discussion

This thesis has presented some background and review of the state of the art in trust and reputation systems based on literature in this field of study. In doing so, a number of outstanding issues in the current trust and reputation models have been identified, particularly pertaining to the suitability of existing systems for agents that are new to a particular e-commerce system or that have little prior experience in the system. To address these problems, the *seniority metric* has been introduced and presented in the context of a full trust model. Finally, the value this metric offers has been evaluated through experimental analysis.

The research here, supported by the experimental analysis, has demonstrated clearly that the seniority metric offers significant advantages in several areas of reputation modeling.

The following discussion presents the value of the seniority metric in each of the areas that it was designed to improve.

6.1 Initializing the Advisor List

A significant oversight in most trust and reputation systems is the lack of a concise means of initializing the set of advisors that an agent consults when making trust decisions. As discussed in section 4.1.1, this problem significantly increases the risk of engaging in transactions for new agents in the e-commerce system. To address this issue, a method of initializing the advisor list by using the seniority metric to identify agents who have been active members of the MAS for a relatively long time is detailed in section 4.4.1.

The performance of models that implement this initialization technique has been evaluated through the four experiments presented in section 5.5. In all four of these experiments, the reputation models relying on the seniority metric performed substantially better than the random selection-based models, particularly over the course of the first ten purchases performed by buyer agents. This indicates that the seniority metric contributes to an increase in performance for new agents in the e-commerce system that have no prior experience on which to base their advisor selection.

The Seniority Base Model used as a test reputation model in the experimental evaluation was specifically created to isolate the effect of advisor list initialization. The results obtained show that in their first ten purchases buyer agents using this model consistently selected *good* vendors more reliably than the Random Base Model and the Random Model. This has been found to be true against scenarios even in conditions where the majority of advisors are dishonest or where bad vendors fill the system.

It has thus been shown that the use of the seniority metric to initialize the advisor list is an effective means of reducing the risk to new agents while enabling them to attain greater results sooner.

6.2 Reducing the experience required

As discussed in section 4.1.2, most reputation models rely heavily on an agent's past experience with vendors and advisors alike. Dependence on direct trust ratings and the use of past experience, or on time and experience, to evolve the set of advisors into a credible set are common among reputation systems. These approaches leave the casual user of the system extremely vulnerable. The seniority model aims to resolve these shortcomings by introducing a metric that is based on parameters that are available even without past personal interaction.

The utility of the seniority metric in this regard is evaluated specifically by the experiment #4: Casual Users, presented in section 5.5.4. The results of the experiment show that the seniority metric benefits buyers immediately upon entering the e-commerce system, as evidenced by the performance of seniority-based models over the first ten purchases. As well, the results show that the agents using the Seniority Model perform significantly better over all purchases. Hence, it can be noted that the use of the seniority to improve the performance of users who transact infrequently is valid.

6.3 Evaluating the credibility of advisors

The mechanisms for assessing the credibility of advisor agents included in existing reputation models rely heavily on past experience. Both new agents and casual user agents are therefore at a significant disadvantage and risk engaging in transactions based on

unreliable information from advisors. The seniority reputation model presented in section 4.4 includes several measures to apply seniority for the assessment of advisor credibility, including the computation used to weight ratings and the technique applied to periodically revise the list of advisors.

Experiment #2: Dishonest Advisors specifically examines the impact of using the seniority metric when the large majority of advisors in the system are dishonest. The results show that the Seniority Model outperforms the other test models, as it enables buyer agents to assess the credibility of agents from the time the buyer agent first enters the system. In particular, over the first ten purchases, agents make significantly improved decisions when employing seniority to select advisors and weight their ratings as evidenced by the results obtained by the Seniority Base and Seniority models. Over all purchases, the performance of the Seniority Base Model falls below that of the Random Model, while the Seniority remains the top performer. This indicates that over time, it is the use of the seniority metric to rate the credibility of advisors and refine the advisor list that protects the buyer agent against dishonest agents, even when dishonest agents dominate the environment.

It has therefore been shown that the seniority metric offers considerable advancement in the area of judging advisor credibility.

6.4 Rating novice agents fairly

Section 4.1.4 introduces the problem of rating novice agents fairly, where new vendors in the e-commerce system are, in some models, assigned similar ratings to other vendors who have a long history of providing poor service. This is an important issue because a new vendor risks being starved of business if its ratings can't compete with a competitors' rating. Since the seniority model introduces a metric based on age and transaction count, it is

proposed that this value can be used to identify directly whether the vendor is a new or established member of the e-commerce community. Additionally, the seniority model makes use of the vendor's seniority rating as a factor when computing the total trust value. This application of the vendor seniority is meant to exaggerate the ratings of senior agents while moderating the ratings of novice agents.

The experiments presented in section 5.5 examine the outcomes of certain scenarios from the buyers' point of view, but do not consider the vendor's perspective. As a result, these experiments cannot be used to prove the hypothesis that the seniority metric makes it possible to fairly rate novice vendor agents so that they can successfully compete with established vendors. While it is strongly believed to be the case, additional experimental development and analysis are required to validate the theory.

6.5 Minimizing the influence of malicious agents

There are several ways, as explained in section 4.1.5, that agents can be malicious within an e-commerce system. In particular, two techniques have been discussed in this thesis: identity changing and ballot stuffing. By depending on agent age and the number of transactions completed through the seniority metric, the seniority reputation model offers protection against these cases. First, if an agent leaves and reenters the system, their seniority is reset to zero and so any clout the agent once had is eliminated; therefore, the benefit of identity changing is removed. Secondly, tactics such as ballot stuffing are discouraged since the total number of transactions factors into seniority. This information can be used to identify and handle suspicious cases.

The results of experiments #2: Dishonest Advisors and #3: Bad Vendors show promising results for the use of the seniority metric for thwarting off devious agents. Some of the

positive results attained in these experiments can be attributed to the techniques described above. However, further experimental analysis designed to specifically verify the effect of the seniority metric in these situations is required to absolutely confirm the contribution believed to be made by the seniority metric in this area.

6.6 Summary

The seniority metric has been developed after thorough review and investigation into the state of the art in reputation modeling. Supported by evidence of equivalent parameters in centralized systems and psychological research into the effect of seniority in groups, the seniority metric and the sample reputation model to employ it have been devised to address shortcomings in existing reputation modeling approaches. Experimental analysis into the affect of the seniority metric in various scenarios has validated the contributions it offers to the field of reputation modeling research. In particular, this research has demonstrated that the seniority metric can be used in advisor list initialization, rating aggregation and advisor list revision to improve the results attained by buyer agents in e-commerce, especially those with little or no prior experience in the subject MAS. Additionally, although further experimental analysis is warranted for validation, strong indications that the seniority metric can be used to more fairly rate novice agents and to deflect the devious attempts of malicious agents have also been presented.

Chapter 7

Conclusions and Future Work

"Trust is central to all transactions" [1]. This phrase is a constant reminder of the importance of a robust trust and reputation model in an e-commerce system. If a user's first interaction with the e-commerce system is a negative one because they were inadequately equipped to make an accurate trust judgment, they are likely to shun the system. Likewise, the trust system must be dependable for the majority of users who, given the break-down of eBay users [24], use the system very casually. Otherwise, trust in the entire system will falter and the viability of the e-commerce MAS will be in question.

Even given the vast amount of research performed in this area, a number of outstanding issues have been identified. By comparing the differences in the parameters used by human individuals in their trust decisions with the parameters used in reputation models for multi-agent systems, the research presented in this thesis has employed a unique approach to addressing those problems.

Through this work, it has been shown that several considerable advancements have been achieved, specifically:

- specification and experimental validation of the use of the seniority metric to improve buyer agents' performance when they are new to the e-commerce system;
- definition and experimental verification of methods to apply the seniority metric to reduce the amount of prior experience required in the e-commerce to make effective trust decisions;
- description of a model for assessing the credibility of advisor agents on an ongoing basis, even with little experience so that all agents produce improved results overall; and
- explanation of the use of the seniority metric to fairly rate novice agents and to defend against malicious agents.

The merits of the *seniority metric* under various scenarios have been demonstrated through experimental analysis. In particular, it has been shown that the seniority metric is effective in environments where good and bad advisors and vendors are equally represented, where dishonest advisors dominate, where bad vendors are the majority and where most of the buyers are casual users of the system. The contribution of a new metric that is computationally simple to implement, yet robust and effective, is an important advancement in the field of reputation modeling for autonomous software agents. Ultimately, the significance of this contribution is highlighted by the importance of achieving robust trust and reputation systems for e-commerce.

7.1 Future Work

The purpose of the research presented has been to define and validate a new metric, seniority, so that it can be applied to new or enhanced trust and reputation models. Given the results of the research, and the validation of the seniority as a metric in trust and reputation systems, further work based on the seniority metric is warranted.

7.1.1 Improving the Seniority Model

First, the seniority trust model presented here can be extended in order to be improved and for further investigation. One such improvement includes having the agent store its own seniority with each internal rating, then use this seniority rating as a weighting factor in the computation of total trust. This would have the effect of aging old internal ratings, while putting more emphasis on more recent ratings. The expected result is a more reliable internal trust component to the system. Additionally, alternate equations to compute seniority, for example by employing a transaction count threshold, could be investigated and examined. As well, development of specific protocols for securely exchanging information required to implement use of the seniority metric within particular MAS systems can be pursued.

Further research into the application of the seniority metric in rating novice vendor agents is also suggested. Such work would be comprised of more experimental evaluation of the present model as well as further development and refinement of techniques to develop fair ratings given the seniority and aggregate reputation rating of a vendor agent.

Finally, the seniority approach can be further advanced through the examination of its use for protecting against malicious behavior. As shown, the model presented here has included provisions for defending against devious advisors and vendor agents. However, this aspect of the seniority model can be elaborated upon. For example, given the parameters of trust,

agent lifetime and transaction count, an activity frequency can be computed. By analyzing this value, especially its value as compared with other agents and its variation over time for a particular agent, patterns of deceitful behavior may be detected. This approach can be validated through its formalization, implementation and subsequent experimental evaluation.

7.1.2 Enhancing Other Reputation Models

Investigation into the application of the seniority metric to other reputation systems is strongly recommended. The relative simplicity of the metric, its parameters and the associated computations facilitate its use to improve other existing reputation models or as a parameter in new enhanced reputation models. H-Trust, as proposed in [37] and [31], is a good candidate since it already includes, for example, a Local Service History Table and a Local Credibility Rating Table to which the seniority of vendors and advisors, respectively could be stored alongside the current rating information. Furthermore, the seniority metric can be applied to reputation models such as [19] that provide mechanisms for evaluating trust from a multi-context or multi-faceted point of view. By applying the seniority metric to models such as these, the amount of improvement that it can offer can be assessed.

7.1.3 Developing New Metrics

Lastly, by continuing the approach initiated here, new and different metrics can be defined based on inspiration from highly successful e-commerce systems like eBay, Amazon and others. Psychological research into community organizational behavior, trust and other related topics can also spark new parameter ideas. For example, in [33] the psychology researchers identified familiarity as a determinant of trust. Familiarity could be represented via a third party in a trust decision. That is, if buyer agent *b* would like advice about *v* but does not have any advisors it can rely on, it can find an advisor *a* that has advice about a

second vendor v_1. By comparing the rating that a has for $v_1$with its own rating for v_1, the agent can determine how similar a is to itself. Using this tactic fosters in b a sense of familiarity with advisor a. This approach is supported by the popular centralized systems where the listings of other entities that a particular individual has rated are posted for other buyers to view.

By pursuing these endeavors, employing the techniques used here to identify other new metrics and by implementing new reputation models that employ the seniority metric, innovative, improved and robust reputation systems can emerge. Consequently, the principles and results demonstrated here can be used to enhance the viability of distributed, open e-commerce systems in general.

References

[1] Partha Dasgupta, "Trust as a Commodity," in *Trust: Making and Breaking Cooperative Relations*, Diego Gambetta, Ed.: Department of Sociology, University Oxford, 2000, pp. 49-72.

[2] Diego Gambetta, "Can We Trust Trust?," in *Trust: Making and Breaking Cooperative Relations*, 1988, pp. 213-237.

[3] Alfarez Abdul-Rahman and Stephen Hailes, "Supporting Trust in Virtual Communities," in *Proceedings of the 33rd Hawaii Internattional Converence on System Sciences*, vol. 6, Washington, DC, USA, 2000, p. 6007.

[4] Alon Altman and Moshe Tennenholtz, "An axiomatic approach to personalized ranking systems," *J. ACM*, vol. 57, no. 4, May 2010, pp. 26:1-26:35.

[5] K Barber and Joonoo Kim, "Belief Revision Process Based on Trust: Agents Evaluating Reputation of Information Sources," *Trust in Cyber-societies, Trust in Cyber-societies*, vol. 2246, 2001, pp. 73-82.

[6] Alberto Caballero, Juan Botia, and Antonio Gomez-Skarmeta, "A New Model for Trust and Reputation Management with an Ontology Based Approach for Similarity Between Tasks," *Multiagent System Technologies, Lecture Notes in Computer Science*, vol. 4196, 2006, pp. 172-183.

[7] R Chohen, K Regan, and T Tran, "Sharing Models of Sellers amongst Buying Agents in Electronic Marketplaces," in *Proceedings of the 10th International Conference on User Modeling—Workshop on Decentralized, Agent Based and Social Approaches to User Modeling*, 2005.

[8] Ghada Derbas, Ayman Kayssi, Hassan Artail, and Ali Chehab, "TRUMMAR - A Trust Model for Mobile Agent Systems Based on Reputation," in *Proceedings of the The IEEE/ACS International Conference on Pervasive Services, ICPS '04*, 2004, pp. 113-120.

[9] Chung-wei Hang, Yonghong Wang, and Munindar P Singh, "An adaptive probabilistic trust model and its evaluation," in *International Converence on Autonomous Agents*, Richland, SC, USA, 2008, pp. 1485-1488.

[10] A Hnativ and S A Ludwig, "Evaluation of trust in an ecommerce multi-agent system using fuzzy reasoning," in *Proceedings of the 18th international conference on Fuzzy Systems, FUZZ-IEEE'09*, Jeju Island, Korea, 2009, pp. 757-763.

[11] TD Huynh, N R Jennings, and N Shadbolt, "Developing an integrated trust and reputation model for open multi-agent systems," in *7th International Workshop on Trust in Agent Societies*, 2004, pp. 65-74.

[12] Audun Josang and Roslan Ismail, "The Beta Reputation System," in *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[13] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th international conference on World Wide Web, WWW '03*, Budapest, Hungary, 2003, pp. 640-651.

[14] Babak Khosravifar, Jamal Bentahar, and Maziar Gomrokchi, "Declarative and Numerical Analysis of Edge Creation Process in Trust-Based Social Networks," *Declarative Agent Languages and Technologies VII, Lecture Notes in Computer Science*, vol. 5948, 2010, pp. 137-157.

[15] S Lee, R Sherwood, and Bobby Bhattacharjee, "Cooperative peer groups in NICE," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, 2003, pp. 1272-1282.

[16] Zhengqiang Liang and Weisong Shi, "PET: A Personalized Trust Model with Reputation and Risk Evaluation for P2P Resource Sharing," in *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, 2005, p. 201b.

[17] Xuelian Long and J Joshi, "BaRMS: A Bayesian Reputation Management approach for P2P systems," in *Information Reuse and Integration (IRI), 2010 IEEE International Conference on*, 2010, pp. 147-152.

[18] E M Maximilien and Munindar P Singh, "Agent-based trust model involving multiple qualities," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, AAMAS '05*, 2005, pp. 519-526.

[19] Steven Reece, Alex Rogers, Stephen Roberts, and Nicholas R Jennings, "Rumours and reputation: evaluating multi-dimensional trust within a decentralised reputation system," in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, MAS '07*, Honolulu, Hawaii, 2007, pp. 165:1-165:8.

[20] Sebastian Ries, "Certain trust: a trust model for users and agents," in *Proceedings of the 2007 ACM symposium on Applied computing, SAC '07*, Seoul, Korea, 2007, pp. 1599-1604.

[21] Jordi Sabater and Carles Sierra, "REGRET: reputation in gregarious societies," in *Proceedings of the fifth international conference on Autonomous agents*, Montreal, Quebec, Canada, 2001, pp. 194-195.

[22] Amirali Salehi-Abari and Tony White, "Towards con-resistant trust models for distributed agent systems," in *Proceedings of the 21st international jont conference on Artifical intelligence*, Pasadena, California, USA, 2009, pp. 272-277.

[23] A A Selcuk, E Uzun, and M R Pariente, "A reputation-based trust management system for P2P networks," in *Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, CCGRID '04*, 2004, pp. 251-258.

[24] S Song, K Hwang, R Zhou, and Y Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," *Internet Computing, IEEE*, vol. 9, nov.-dec. 2005, pp. 24-34.

[25] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," *Autonomous Agents and Multi-Agent Systems*, vol. 12, no. 2, March 2006, pp. 183-198.

[26] A Vizcaino, J Portillo-Rodriguez, J P Soto, M Piattini, and O Kusche, "A recommendation algorithm for Knowledge Objects based on a trust model," in *Research Challenges in Information Science, 2009. RCIS 2009.*, 2009, pp. 93-102.

[27] Li Xiong and Ling Liu, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *Knowledge and Data Engineering, IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, July 2004, pp. 843-857.

[28] Bin Yu, Munindar P Singh, and Katia Sycara, "Developing Trust in Large-Scale Peer-to-Peer Systems," in *Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability*, 2004, pp. 1-10.

[29] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes, "Collaborative reputation mechanisms for electronic marketplaces," *Decision Support Systems*, vol. 29, pp. 371-388, 2000.

[30] Giorgos Zacharia, "Trust management through reputation mechanisms," *Applied Artificial Intelligence*, vol. 14, 2000, pp. 881-907.

[31] Huanyu Zhao and Xiaolin Li, "H-trust: a group trust management system for pee-to-peer desktop grid," *J. Comput. Sci. Technol.*, vol. 24, no. 5, September 2009, pp. 833-843.

[32] Runfang Zhou and Kai Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, April 2007, pp. 460-473.

[33] Chester A Insko et al., "Seniority in the generational transition of laboratory groups: The effects of social familiarity and task experience," *Journal of Experimental Social Psychology*, vol. 18, no. 6, 1982, pp. 557-580.

[34] Alessandro Garcia et al., "Software engineering for large-scale multi-agent systems:

SELMAS 2003: workshop report," *SIGSOFT Softw. Eng. Notes*, vol. 28, no. 6, November 2003, pp. 1-1.

[35] Anish Biswas, Sandip Debnath, Sandip Sen, Ip Debnath, and Ip Sen, "Believing Others: Pros and Cons," *Artificial Intelligence*, vol. 142, 2000, pp. 276-286.

[36] Elizabeth Sklar and Mathew Davies, "SimEd: Simulating Education as a Multi Agent System," *Autonomous Agents and Multiagent Systems, International Joint Conference on*, vol. 3, 2004, pp. 998-1005.

[37] Huanyu Zhao and Xiaolin Li, "H-Trust: A Robust and Lightweight Group Reputation System for Peer-to-Peer Desktop Grid," in *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on Distributed Computing Systems Workshops*, 2008, pp. 235-240.

[38] Katia Sycara, "Multiagent Systems," *AI Magazine*, vol. 10, no. 2, 1998, pp. 79-93.

[39] Sini Ruohomaa, Lea Kutvonen, and Eleni Koutrouli, "Reputation Management Survey," in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, 2007, pp. 103 -111.

[40] Bin Yu and Munindar P Singh, "An evidential model of distributed reputation management," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, AAMAS '02*, Bologna, Italy, 2002, pp. 294--301.

[41] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt, "Notions of reputation in multi-agents systems: a review," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, AAMAS '02*, Bologna, Italy, 2002, pp. 280-287.

[42] Yonghong Wang and Munindar P Singh, "Trust representation and aggregation in a distributed agent system," in *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, 2006, pp. 1425-1430.

[43] (2011, March) eBay. [Online]. HYPERLINK "http://pages.ebay.com/help/feedback/scores-reputation.html" http://pages.ebay.com/help/feedback/scores-reputation.html

[44] (2011, April) Amazon Help: Customer Reviews. [Online]. HYPERLINK "http://www.amazon.com/gp/help/customer/display.html/" http://www.amazon.com/gp/help/customer/display.html/

[45] (2011, April) Epinions.com - FAQs: Product Ratings. [Online]. HYPERLINK "http://www99.epinions.com/help/faq/?show=faq_prodrate" http://www99.epinions.com/help/faq/?show=faq_prodrate

[46] (2011, April) Apple - Support - Mac App Store - Get Started. [Online]. HYPERLINK

"http://www.apple.com/support/mac/app-store/getstarted/" http://www.apple.com/support/mac/app-store/getstarted/

[47] Wang Shixian and Wei Siying, "Research on multi-agent system based trust model of partner selection of virtual enterprise," in *First International Conference on Networked Digital Technologies, 2009. NDT '09.* , 2009, pp. 512-514.

[48] Zhigang Huang, Pengyi Gao, Yan He, and Qing He, "Multi-Agent Cooperation Based on Interest Group," in *International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009.*, 2009, pp. 1-5.

[49] A Gutowska and K Buckley, "A Computational Distributed Reputation Model for B2C E-commerce," in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT '08* , vol. 3, 2008, pp. 72-76.

[50] Fengming Liu et al., "DynamicTrust: three-dimensional dynamic computing model of trust in peer-to-peer networks," in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC '09*, Shanghai, China, 2009, pp. 337-344.

[51] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, "The PageRank Citation Ranking: Bringing Order to the Web.," Stanford InfoLab, Technical Report 1999.

[52] Thomas Tran and Robin Cohen, "Improving User Satisfaction in Agent-Based Electronic Marketplaces by Reputation Modelling and Adjustable Product Quality," , 2004, pp. 828-835.

[53] (2011, April) Nikon D90 Digital Camera with 18-105mm lens Revews. Buying Guides & Consumer Product Reviews - Epinions.com. [Online]. HYPERLINK "http://www1.epinions.com/reviews/Nikon_D90_Digital_Camera_with_18_105mm_lens"http://www1.epinions.com/reviews/Nikon_D90_Digital_Camera_with_18_105mm_lens

[54] (2011, April) Epinions.com - aloofyouth's profile. [Online]. HYPERLINK "http://www1.epinions.com/user-aloofyouth" http://www1.epinions.com/user-aloofyouth

[55] (2011, April) New Nikon D90 D-90 Camera w/ 18-55 mm VR 3Lens 8GB USA | eBay. [Online]. HYPERLINK "http://cgi.ebay.com/New-Nikon-D90-D-90-Camera-w-18-55mm-VR-3Lens-8GB-USA-/140528920314?pt=Digital_Cameras&hash=item20b82d26fa" http://cgi.ebay.com/New-Nikon-D90-D-90-Camera-w-18-55mm-VR-3Lens-8GB-USA-/140528920314?pt=Digital_Cameras&hash=item20b82d26fa

[56] (2011, April) eBay Feedback Profile for bigeasycameras. [Online]. HYPERLINK "http://feedback.ebay.com/ws/eBayISAPI.dll?ViewFeedback2&userid=bigeasycameras&ftab=AllFeedback&myworld=true"

http://feedback.ebay.com/ws/eBayISAPI.dll?ViewFeedback2&userid=bigeasycameras&ftab=AllFeedback&myworld=true

[57] (2011, April) eBay Feedback Profile for soxwin0407. [Online]. HYPERLINK "http://feedback.ebay.com/ws/eBayISAPI.dll?ViewFeedback2&userid=soxwin0407&ftab=AllFeedback&myworld=true" http://feedback.ebay.com/ws/eBayISAPI.dll?ViewFeedback2&userid=soxwin0407&ftab=AllFeedback&myworld=true

[58] Stefan Spitz and York Tüchelmann, "A Trust Model Considering the Aspects of Time," in *Proceedings of the 2009 Second International Conference on Computer and Electrical Engineering - Volume 01, ICCEE '09*, 2009, pp. 550-554.

[59] Bin Yu and Munindar P Singh, "A Social Mechanism of Reputation Management in Electronic Communities," in *Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, 2000, pp. 154-165.

[60] (2010) Oxford Reference Online. [Online]. HYPERLINK "http://www.oxfordreference.com.proxy.bib.uottawa.ca/views/BOOK_SEARCH.html?subview=BookHomePage&book=t140" http://www.oxfordreference.com.proxy.bib.uottawa.ca/views/BOOK_SEARCH.html?subview=BookHomePage&book=t140

[61] Nathan Griffiths, "Task delegation using experience-based multi-dimensional trust," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, AAMAS '05*, The Netherlands, 2005, pp. 489-496.

[62] Jeffrey Tweedale and Philip Cutler, "Trust in Multi-Agent Systems," in *Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science Series*, Bogdan Gabrys, Robert Howlett, and Lakhmi Jain, Eds.: Springer Berlin / Heidelberg, 2006, vol. 4252, pp. 479-485.

[63] Selmas W Report et al., "Software Engineering for Large-Scale Multi-Agent Systems," 2002.

[64] Catherine Cormier and Thomas T Tran, "Improving Trust and Reputation Modeling in E-Commerce Using Agent Lifetime and Transaction Count," *E-Technologies: Innovation in an Open World, Lecture Notes in Business Information Processing*, vol. 26, 2009, pp. 184-195.

[65] Sarvapali D Ramchurn, Dong Huynh, and Nicholas R Jennings, "Trust in multi-agent systems," *Knowl. Eng. Rev.*, vol. 19, no. 1, March 2004, pp. 1-25.

Appendix A

Sample Source Code

As a sample of the software simulation used in the experimental analysis, source code is provided for:

- *Advisor* Interface;
- *BuyingAgent* Class;
- *ReputationBuyingAgent* Class (Random Base Model);
- *SeniorityBuyingAgent* Class (Seniority Base Model);
- *SeniorityAndDirectBuyingAgent* Class; amd
- *SeniorityDirectUpdateAdvisorsAgent* Class (Seniority Model).

Advisor Interface

```
package trustsim;

import java.util.Hashtable;

import trustsim.buyer.BuyingAgent;

/**
 * The interface implemented by all buyer agents so that they can act as
 * advisors to other agents.
 *
 */

public interface Advisor {

	public Double getRating(SellingAgent a);

	Hashtable<BuyingAgent, Double> getTopAdvisors(int n);
}
```

BuyingAgent Class

```
/**
 * This class is the super-class of all buying agents; buying agents inherit
 * their activity level control and main buying algorithm from this class.
 *
 * It implements the Advisor interface.
 *
 */

package trustsim.buyer;

import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import trustsim.Advisor;
import trustsim.Agent;
import trustsim.Marketplace;
import trustsim.Purchase;
import trustsim.SellingAgent;
import trustsim.Simulator;
import trustsim.Utility;

public class BuyingAgent extends Agent implements Advisor {

	public static final int HONEST_ADVISOR = 0;
	public static final int DISHONEST_ADVISOR_HIGH = 1;
	public static final int DISHONEST_ADVISOR_LOW = 2;
	public static final int DISHONEST_ADVISOR_ERRATIC = 3;

	public static final int ACTIVITY_CONSTANT = 0;
	public static final int ACTIVITY_HIGH = 1;
	public static final int ACTIVITY_MEDIUM = 2;
	public static final int ACTIVITY_LOW = 3;
	public static final int ACTIVITY_VERY_LOW = 4;

	private Purchase m_purchase = null;
	private Hashtable<Integer, Vector<Double>> m_ratings = new Hashtable<Integer,
Vector<Double>>();
```

```
private int m_advisorType = HONEST_ADVISOR;
private int m_groupNumber = 0;
private int m_activityLevel = ACTIVITY_CONSTANT;
private int m_stepsTilBuy = 0;

public BuyingAgent(int id) {
	super(id);
}

public BuyingAgent(int id, int group) {
	super(id);
	m_groupNumber = group;
}

public BuyingAgent(int id, int group, int type, int activityLevel) {
	super(id);
	m_groupNumber = group;
	m_advisorType = type;
	m_activityLevel = activityLevel;
}

public void clearResult() {
	m_purchase = null;
}

public Purchase getPurchase() {
	return m_purchase;
}

public int getGroupNumber() {
	return m_groupNumber;
}

public void setActivityLevel(int l) {
	m_activityLevel = l;
}

public void buy() {
	//determine if it is time to buy based on activity level
	if (m_stepsTilBuy == 0) {
		private_buy();
		setStepsTilBuy();
	} else
		m_stepsTilBuy--;
}

public void private_buy() {
	Marketplace market = getMarketplace();
	if (market != null) {
		prepareToBuy();
		List<SellingAgent> sellers = market.getSellers();
		SellingAgent seller = chooseSeller(sellers);
		if (seller != null) {
			int value = seller.sell();
			m_purchase = new Purchase(seller, value);
			updateRatings(seller, value);
			addTransaction();
		}
	}
}

public boolean hasPurchased() {
	return (m_purchase != null);
}

SellingAgent chooseSeller(List<SellingAgent> sellers) {
	// random selection from all sellers in marketplace
	return randomlyChooseSeller(sellers);
}

SellingAgent randomlyChooseSeller(List<SellingAgent> sellers) {
```

```
        int n = sellers.size() - 1;
        int i = (int) Math.round(Math.random() * n);
        return sellers.get(i);
}

void updateRatings(SellingAgent seller, int value) {
        Integer sellerId = new Integer(seller.getId());
        Double rating = new Double(((double) value) - 5) / 5;
        if (m_ratings == null)
                m_ratings = new Hashtable<Integer, Vector<Double>>();
        Vector<Double> r;
        if (m_ratings.containsKey(sellerId)) {
                r = m_ratings.get(sellerId);
                r.add(rating);
        } else {
                r = new Vector<Double>();
                r.add(rating);
        }
        m_ratings.put(sellerId, r);
}

Double getHonestRating(Agent a) {
        Double honestRating = null;
        if (m_ratings != null) {
                Integer targetId = a.getId();
                Vector<Double> ratings = m_ratings.get(targetId);
                if (ratings != null) {
                        double r = 0;
                        Iterator<Double> rList = ratings.iterator();
                        while (rList.hasNext()) {
                                r += rList.next().doubleValue();
                        }
                        int n = ratings.size();
                        honestRating = (n > 0) ? r / n : null;
                }
        }
        return honestRating;
}

protected Double generateRating(double low, double high) {
        return Utility.randomFromRange(low, high);
}

private void setStepsTilBuy() {
        switch (m_activityLevel) {
        case ACTIVITY_CONSTANT:
                m_stepsTilBuy = 0;
                break;

        case ACTIVITY_HIGH:
                m_stepsTilBuy = Utility.randomFromRange(0, 2);
                break;

        case ACTIVITY_MEDIUM:
                m_stepsTilBuy = Utility.randomFromRange(3, 6);
                break;

        case ACTIVITY_LOW:
                m_stepsTilBuy = Utility.randomFromRange(7, 9);
                break;

        case ACTIVITY_VERY_LOW:
                m_stepsTilBuy = 10;
                break;
        }
}

public String getActivityLevelString() {
        String s = "invalid";
        switch (m_activityLevel) {
        case ACTIVITY_CONSTANT:
```

```
                s = "constant";
                break;

        case ACTIVITY_HIGH:
                s = "high";
                break;

        case ACTIVITY_MEDIUM:
                s = "medium";
                break;

        case ACTIVITY_LOW:
                s = "low";
                break;

        case ACTIVITY_VERY_LOW:
                s = "very low";
                break;
        }
        return s;
}

public int getAdvisorType() {
        return m_advisorType;
}

public String getAdvisorTypeString() {
        String s = "invalid";
        switch (m_advisorType) {
        case HONEST_ADVISOR:
                s = "honest";
                break;
        case DISHONEST_ADVISOR_HIGH:
                s = "dishonest-high";
                break;
        case DISHONEST_ADVISOR_LOW:
                s = "dishonest-low";
                break;
        case DISHONEST_ADVISOR_ERRATIC:
                s = "dishonest-irratic";
                break;
        }
        return s;
}

public String toString() {
        return "" + this.getId() + "," + this.getGroupNumber() + ","
                        + this.getTimestamp() + "," + this.getActivityLevelString()
                        + "," + this.getAdvisorTypeString() + ", "
                        + Simulator.getTypeString(Simulator.getBuyerType(this));
}

public static String getHeaderString() {
        return "ID,Group,Timestamp,Activity Level,Advisor Type";
}

Hashtable<Integer, Vector<Double>> getRatings() {
        return m_ratings;
}

void prepareToBuy() {
        // Do nothing
}

/*** Advisor Methods **/

public Double getRating(SellingAgent a) {
        Double rating = null;
        switch (m_advisorType) {
        case HONEST_ADVISOR:
                rating = getHonestRating(a);
```

```
                        break;
                case DISHONEST_ADVISOR_HIGH:
                        rating = generateRating(0, 1);
                        break;
                case DISHONEST_ADVISOR_LOW:
                        rating = generateRating(-1, 0);
                        break;
                case DISHONEST_ADVISOR_ERRATIC:
                        rating = generateRating(-1, 1);
                        break;
                }
                return rating;
        }

        public Hashtable<BuyingAgent, Double> getTopAdvisors(int n) {
                return null;
        }

}
```

ReputationBuyingAgent Class

```
/**
 * This class is the Random Base Model.
 * Sellersa are selected strictly based on advice from
 * advisors randomly chosen on the first buy.
 *
 * It extends the class BuyingAgent
 *
 */

package trustsim.buyer;

import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.List;
import java.util.Set;
import java.util.Vector;

import trustsim.Agent;
import trustsim.SellingAgent;

public class ReputationBuyingAgent extends BuyingAgent {

        static final int ADVISOR_LIST_SIZE = 10;
        static final double TRUST_THRESHOLD = 0.75;

        private List<BuyingAgent> m_advisors;
        Hashtable<BuyingAgent, Vector<Rating>> m_advisor_ratings;
        protected Vector<Rating> m_lastAdvice;

        public ReputationBuyingAgent(int id) {
                super(id);
        }

        public ReputationBuyingAgent(int id, int group) {
                super(id, group);
        }

        public ReputationBuyingAgent(int id, int group, int type, int activityLevel) {
                super(id, group, type, activityLevel);
        }

        public void buy() {
                if (m_advisors == null) {
                        m_advisors = selectAdvisors(ADVISOR_LIST_SIZE);
                }
                super.buy();
        }

        SellingAgent chooseSeller(List<SellingAgent> sellingAgents) {
                if (m_advisors == null) {
                        m_advisors = selectAdvisors(ADVISOR_LIST_SIZE);
                }
                return chooseSeller(sellingAgents, m_advisors);
        }

        SellingAgent chooseSeller(List<SellingAgent> sellingAgents,
                        List<BuyingAgent> advisors) {
                SellingAgent seller = null;

                // pick the agent with the highest reputation
                SellerRatingTable ratingTable = new SellerRatingTable();
                Hashtable<Integer, Vector<Rating>> allAdviceTable = new Hashtable<Integer,
Vector<Rating>>();

                for (SellingAgent s : sellingAgents) {
                        Vector<Rating> adviceVector = new Vector<Rating>();
```

```
                double r = getAdvice(advisors, s, adviceVector);
                Double totalRating = (r < -1 || r > 1) ? null : r;
                ratingTable.add(s, totalRating);
                allAdviceTable.put(s.getId(), adviceVector);
        }

        seller = ratingTable.getMaxRatedSellingAgent(TRUST_THRESHOLD);
        m_lastAdvice = allAdviceTable.get(new Integer(seller.getId()));

        return seller;
}

List<BuyingAgent> getAdvisors() {
        return m_advisors;
}

void setAdvisors(List<BuyingAgent> advisorList) {
        m_advisors = advisorList;
}

List<BuyingAgent> selectAdvisors(int maxSize) {
        return selectAdvisors(maxSize, null);
}

List<BuyingAgent> selectAdvisors(int maxSize, List<BuyingAgent> blacklist) {
        // get all possible advisors
        List<BuyingAgent> advisorAgents = getMarketplace().getBuyers();
        int n_advisors = advisorAgents.size() - 1;
        if (n_advisors > maxSize)
                n_advisors = maxSize;

        Vector<BuyingAgent> advisors = new Vector<BuyingAgent>();

        // randomly choose advisors
        while (advisors.size() < n_advisors) {
                int n = advisorAgents.size() - 1;
                int i = (int) Math.round(Math.random() * n);
                BuyingAgent a = advisorAgents.get(i);
                if (a != null && a != this && !advisors.contains(a)
                                && canAddAdvisor(a, blacklist))
                        advisors.add(a);
        }
        return advisors;
}

private boolean canAddAdvisor(BuyingAgent a, List<BuyingAgent> blacklist) {
        boolean canAdd = true;
        if (blacklist != null)
                canAdd = !blacklist.contains(a);
        return canAdd;
}

public String getAdvisorsString() {
        String s = "[";
        Iterator<BuyingAgent> advisors = m_advisors.iterator();
        boolean addcomma = false;
        while (advisors.hasNext()) {
                BuyingAgent a = advisors.next();
                s = s + (addcomma ? ", " : "") + a.getId();
                addcomma = true;
        }
        s = s + "]";
        return s;
}

double getAdvice(List<BuyingAgent> advisorList, SellingAgent s,
                Vector<Rating> adviceVector) {
        double rating = 0;
        int n = 0;
        for (BuyingAgent a : advisorList) {
                Double ra = a.getRating(s);
```

```
                        if (ra != null) {
                                rating += ra.doubleValue();
                                n++;
                                if (adviceVector != null)
                                        adviceVector.add(new Rating(a, s, ra));
                        }
                }

                if (n > 0)
                        rating = rating / n;
                else
                        rating = -2; // return invalid value to indicate no advice was
                                                        // received
                return rating;
        }

        void updateRatings(SellingAgent seller, int value) {
                super.updateRatings(seller, value);
                double myRating = (value - 5.0) / 5.0;
                if (m_lastAdvice != null) {
                        for (Rating r : m_lastAdvice) {
                                updateAdvisorRatings(r, myRating);
                        }
                }
        }

        void updateAdvisorRatings(Rating r, double myRating) {
                Rating advisorRating = new Rating(new Double(1 - Math.abs(r.getRating()
                                - myRating)));
                Agent a = r.getRater();
                if (a != null) {
                        if (a instanceof BuyingAgent) {
                                BuyingAgent advisor = (BuyingAgent) a;
                                Vector<Rating> ratingVector = null;

                                if (m_advisor_ratings == null) {
                                        m_advisor_ratings = new Hashtable<BuyingAgent,
Vector<Rating>>();
                                } else
                                        ratingVector = m_advisor_ratings.get(advisor);

                                if (ratingVector == null)
                                        ratingVector = new Vector<Rating>();

                                ratingVector.add(advisorRating);
                                m_advisor_ratings.put(advisor, ratingVector);
                        }
                }
        }

        protected void printAdvisorRatings() {
                System.out.println("--------");
                Set<BuyingAgent> advisorSet = m_advisor_ratings.keySet();
                for (BuyingAgent a : advisorSet) {
                        Vector<Rating> ratingVector = m_advisor_ratings.get(a);
                        System.out.print("advisor:" + a.getId() + ", ratings: ");
                        for (Rating r : ratingVector) {
                                if (r == null)
                                        System.out.println("null");
                                else
                                        System.out.print(r);
                        }
                        System.out.print("\n");
                }
                System.out.println("--------");

        }

        Hashtable<BuyingAgent, Double> getTopRatedAdvisors(int n) {
                Hashtable<BuyingAgent, Double> advisorsTable = null;
                switch (getAdvisorType()) {
```

```
                case HONEST_ADVISOR:
                        advisorsTable = getTopRatedAdvisors_Honest(n);
                        break;
                default:
                        advisorsTable = getTopRatedAdvisors_Dishonest(n, getAdvisorType());
                        break;
                }
                return advisorsTable;
        }

        Hashtable<BuyingAgent, Double> getTopRatedAdvisors_Dishonest(int n,
                        int dishonestyType) {
                Hashtable<BuyingAgent, Double> advisorsTable = null;
                List<BuyingAgent> advisorList = selectAdvisors(n);
                for (BuyingAgent a : advisorList) {
                        Double rating = null;
                        switch (dishonestyType) {
                        case DISHONEST_ADVISOR_HIGH:
                                rating = generateRating(0, 1);
                                break;
                        case DISHONEST_ADVISOR_LOW:
                                rating = generateRating(-1, 0);
                                break;
                        case DISHONEST_ADVISOR_ERRATIC:
                                rating = generateRating(-1, 1);
                                break;
                        }
                        advisorsTable.put(a, rating);
                }
                return advisorsTable;
        }

        Hashtable<BuyingAgent, Double> getTopRatedAdvisors_Honest(int n) {
                AdvisorRatingTable advisorRatingTable = new AdvisorRatingTable();

                Enumeration<BuyingAgent> advisorKeys = m_advisor_ratings.keys();
                while (advisorKeys.hasMoreElements()) {
                        BuyingAgent a = advisorKeys.nextElement();
                        Vector<Rating> ratingVector = m_advisor_ratings.get(a);
                        Double totalRating = 0.0;
                        int nRatings = 0;
                        for (Rating r : ratingVector) {
                                if (r != null) {
                                        totalRating += r.getRating();
                                        nRatings++;
                                }
                        }
                        if (nRatings > 0)
                                advisorRatingTable.add(a, totalRating / nRatings);
                }

                return advisorRatingTable.getTopRatedAgents(n, null);
        }

}
```

SeniorityBuyingAgent Class

```
/**
 * This class is the Seniority Base Model.
 * Advisors are chosen based on seniority at the start of
 * the first buy. Advice is then weighted based on seniority.
 *
 * It extends ReputationBuyingAgent.
 *
 */

package trustsim.buyer;

import java.util.Arrays;
import java.util.Comparator;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.List;
import java.util.Vector;

import trustsim.AdvisorSeniority;
import trustsim.Agent;
import trustsim.SellingAgent;

public class SeniorityBuyingAgent extends ReputationBuyingAgent implements
                Comparator<AdvisorSeniority> {

        public static final int METHOD_PRODUCT = 0;
        public static final int METHOD_SUM = 1;

        private int m_method = METHOD_PRODUCT;
        private double m_lifetimeWeight = 0.5;

        private boolean m_useSellerSeniority = false;

        public SeniorityBuyingAgent(int id, int group, int type, int activityLevel,
                        int method) {
                super(id, group, type, activityLevel);
                m_method = method;
        }

        double getAdvice(List<BuyingAgent> advisorList, SellingAgent s,
                        Vector<Rating> adviceVector) {
                double rating = 0;
                int n = 0;
                long t = System.currentTimeMillis();
                float totalSeniority = 0;
                for (BuyingAgent a : advisorList) {
                        Double ra = a.getRating(s);
                        if (ra != null) {
                                float seniority = calculateSeniority(a, t);
                                rating += seniority * ra.doubleValue();
                                n++;
                                totalSeniority += seniority;
                                if (adviceVector != null)
                                        adviceVector.add(new SeniorityRating(a, new Double(
                                                        seniority), s, null, ra));
                        }
                }

                float sellerSeniority = (m_useSellerSeniority) ? calculateSeniority(s,
                                t) : 1;

                if (n > 0 && totalSeniority > 0)
                        rating = sellerSeniority * (rating / totalSeniority);
                else
                        rating = -2; // return invalid value to indicate no advice was
                                                // received
                return rating;
```

```
}

List<BuyingAgent> selectAdvisors(int maxSize) {
	return selectAdvisors(maxSize, null);
}

List<BuyingAgent> selectAdvisors(int maxSize, List<BuyingAgent> blacklist) {
	// get all possible advisors
	List<BuyingAgent> advisorAgents = getMarketplace().getBuyers();
	int n_advisors = advisorAgents.size();
	if (n_advisors > maxSize)
		n_advisors = maxSize;

	long t = System.currentTimeMillis();

	AdvisorSeniority[] seniorityArray = new AdvisorSeniority[advisorAgents
			.size()];
	int i = 0;
	for (BuyingAgent a : advisorAgents) {

		float seniority = calculateSeniority(a, t);

		seniorityArray[i] = new AdvisorSeniority(a, seniority);
		i++;
	}
	Vector<BuyingAgent> advisorList = new Vector<BuyingAgent>();
	int n = seniorityArray.length - 1;
	if (n >= 0) {
		Arrays.sort(seniorityArray, this);
		i = 0;
		while (advisorList.size() < n_advisors && i <= n) {
			BuyingAgent a = seniorityArray[n - i].getAdvisor();
			if (!advisorList.contains(a) && !(a == this)
					&& canUseAdvisor(a, blacklist))
				advisorList.add(a);
			i++;
		}

	}

	return advisorList;
}

private boolean canUseAdvisor(BuyingAgent a, List<BuyingAgent> blacklist) {
	boolean canAdd = true;
	if (blacklist != null)
		canAdd = !blacklist.contains(a);
	return canAdd;
}

public int compare(AdvisorSeniority a1, AdvisorSeniority a2) {
	float diff = a1.getSeniority() - a2.getSeniority();
	int retVal = 0;
	if (diff < 0)
		retVal = -1;
	else if (diff > 0)
		retVal = 1;
	return retVal;
}

float calculateSeniority(Agent a, long t) {
	float s = 0;
	long dob = a.getTimestamp();
	double lifetime = (1.0) - ((double) dob) / t;
	int transactionCount = a.getTransactionCount();
	s = (float) (lifetime * (transactionCount));
	return s;
}

public double getLifetimeWeight() {
	return m_lifetimeWeight;
```

```
	}

	public void setLifetimeWeight(double w) {
		this.m_lifetimeWeight = w;
	}

	public int getMethod() {
		return m_method;
	}

	void updateAdvisorRatings(Rating r, double myRating) {
		if (r != null) {
			if (r instanceof SeniorityRating) {
				Double advisorSeniority = ((SeniorityRating) r)
						.getRaterSeniority();
				Double advisorRating = new Double(1 - Math.abs(r.getRating()
						- myRating)); // advisor rating is based on
difference
                                                                        // between their
seller rating and mine
				Agent a = r.getRater();
				if (a != null) {
					if (a instanceof BuyingAgent) {
						BuyingAgent advisor = (BuyingAgent) a;
						Vector<Rating> ratingVector = null;

						if (m_advisor_ratings == null) {
							m_advisor_ratings = new
Hashtable<BuyingAgent, Vector<Rating>>();
						} else
							ratingVector =
m_advisor_ratings.get(advisor);

						if (ratingVector == null)
							ratingVector = new Vector<Rating>();

						SeniorityRating sr = new SeniorityRating(null,
null,
                                                        advisor, advisorSeniority,
advisorRating);

						ratingVector.add(sr);
						m_advisor_ratings.put(advisor, ratingVector);
					}
				}
			}
		}
	}

	Hashtable<BuyingAgent, Double> getTopRatedAdvisors_Honest(int n) {
		AdvisorRatingTable advisorRatingTable = new AdvisorRatingTable();

		Enumeration<BuyingAgent> advisorKeys = m_advisor_ratings.keys();
		while (advisorKeys.hasMoreElements()) {
			BuyingAgent a = advisorKeys.nextElement();
			Vector<Rating> ratingVector = m_advisor_ratings.get(a);
			Double totalRating = 0.0;
			Double totalSeniority = 0.0;
			for (Rating r : ratingVector) {
				if (r != null) {
					if (r instanceof SeniorityRating) {
						Double sr = ((SeniorityRating)
r).getRateeSeniority();
						totalRating += sr * r.getRating();
						totalSeniority += sr;
					}
				}
			}
			if (totalSeniority > 0)
				advisorRatingTable.add(a, totalRating / totalSeniority);
		}
```

```
                return advisorRatingTable.getTopRatedAgents(n, null);
        }

        public void setUseSellerSeniority(boolean b) {
                m_useSellerSeniority = b;
        }

        public boolean getUseSellerSeniority() {
                return m_useSellerSeniority;
        }

}
```

SenioritыandDirectBuyingAgent Class

```
/**
 * This class extends the class SeniorityBuyingAgent
 * by adding the use of internal trust to compute total
 * trust.
 *
 */

package trustsim.buyer;

import java.util.Hashtable;
import java.util.List;
import java.util.Vector;

import trustsim.SellingAgent;

public class SeniorityAndDirectBuyingAgent extends SeniorityBuyingAgent {

	public SeniorityAndDirectBuyingAgent(int id, int group, int type,
			int activityLevel) {
		super(id, group, type, activityLevel, METHOD_PRODUCT);
	}

	SellingAgent chooseSeller(List<SellingAgent> sellingAgents,
			List<BuyingAgent> advisors) {
		SellingAgent seller = null;

		// pick the agent with the highest reputation
		SellerRatingTable ratingTable = new SellerRatingTable();
		Hashtable<Integer, Vector<Rating>> allAdviceTable = new Hashtable<Integer,
Vector<Rating>>();

		for (SellingAgent s : sellingAgents) {
			Vector<Rating> adviceVector = new Vector<Rating>();
			double r = getAdvice(advisors, s, adviceVector);

			// add the internal rating to the seller reputation
			Double totalRating = addInternalRating(s, r);
			ratingTable.add(s, totalRating);
			allAdviceTable.put(s.getId(), adviceVector);
		}

		seller = ratingTable.getMaxRatedSellingAgent(TRUST_THRESHOLD);
		m_lastAdvice = allAdviceTable.get(new Integer(seller.getId()));

		return seller;
	}

	private Double addInternalRating(SellingAgent s, double r) {
		Double rating = null;
		if (r < -1 || r > 1) { // the reputation value is not valid, so use
                                                    // internal trust only
			rating = getHonestRating(s);
		} else {
			rating = r;
			Double ir = getHonestRating(s);
			if (ir != null) {
				double ir_val = ir.doubleValue();
				if (ir_val >= -1 && ir_val <= 1) {
					//new rating is average of advice and internal rating
					rating = new Double((ir_val + r) / 2);
				}
			}
		}
		return rating;
	}

}
```

SeniorityDirectUpdateAdvisorsAgent Class

```
/**
 * This is the Seniority Model.
 *
 * The class extends the class SeniorityBuyingandDirectAgent
 * to include the ability to revise the advisor list.
 * Seniority is used to judge the credibility of advisors, and
 * new advisors are selected by first requesting recommendations
 * from advisors, then if necessary by using seniority only.
 *
 */

package trustsim.buyer;

import java.util.Hashtable;
import java.util.List;
import java.util.Set;
import java.util.Vector;

import trustsim.Simulator;

public class SeniorityDirectUpdateAdvisorsAgent extends
                SeniorityAndDirectBuyingAgent {

        static final int ADVISOR_REFRESH_CYCLE = Simulator.ADVISOR_REFRESH_CYCLE;
        static final Double ADVISOR_RATING_THRESHOLD = 0.75;

        Vector<BuyingAgent> m_advisorBlacklist = null;
        int m_stepsToRefreshAdvisors = ADVISOR_REFRESH_CYCLE;

        public SeniorityDirectUpdateAdvisorsAgent(int id, int group, int type,
                        int activityLevel) {
                super(id, group, type, activityLevel);
        }

        void prepareToBuy() {
                if (m_stepsToRefreshAdvisors == 0) {
                        Vector<BuyingAgent> unfavorableAdvisors = removeUnfavorableAdvisors(
                                        getAdvisors(), ADVISOR_RATING_THRESHOLD);

                        if (unfavorableAdvisors != null && unfavorableAdvisors.size() > 0) {
                                blacklistAdvisors(unfavorableAdvisors);
                                addNewAdvisors(getAdvisors(), unfavorableAdvisors.size());
                        }

                        m_stepsToRefreshAdvisors = ADVISOR_REFRESH_CYCLE;
                } else
                        m_stepsToRefreshAdvisors--;

        }

        private void addNewAdvisors(List<BuyingAgent> advisorList, int n) {
                // pick the agent with the highest reputation
                AdvisorRatingTable ratingTable = new AdvisorRatingTable();
                long t = System.currentTimeMillis();
                Double totalSeniority = computeTotalSeniority(advisorList, t);
                if (totalSeniority > 0) {
                        for (BuyingAgent a : advisorList) {
                                Double sWeight = calculateSeniority(a, t) / totalSeniority;
                                Hashtable<BuyingAgent, Double> recommendedAdvisors = a
                                                .getTopAdvisors(n);
                                if (recommendedAdvisors != null) {
                                        Set<BuyingAgent> recommendedAdvisorsSet =
recommendedAdvisors
                                                        .keySet();
                                        for (BuyingAgent ba : recommendedAdvisorsSet) {
                                                Double rating = sWeight *
recommendedAdvisors.get(ba);
```

```
                                        ratingTable.add(ba, rating);
                                }
                        }
                }
                Vector<BuyingAgent> cantAddAdvisors = new Vector<BuyingAgent>();
                cantAddAdvisors.addAll(advisorList);
                cantAddAdvisors.addAll(m_advisorBlacklist);

                Hashtable<BuyingAgent, Double> topPicks = ratingTable
                                .getTopRatedAgents(n, cantAddAdvisors);

                Set<BuyingAgent> advisorSet = topPicks.keySet();
                for (BuyingAgent advisor : advisorSet) {
                        advisorList.add(advisor);
                }

                if (topPicks.size() < n) {
                        cantAddAdvisors.clear();
                        cantAddAdvisors.addAll(advisorList);
                        cantAddAdvisors.addAll(m_advisorBlacklist);
                        List<BuyingAgent> moreAdvisorsList = selectAdvisors(n
                                        - topPicks.size(), cantAddAdvisors);
                        if (moreAdvisorsList != null)
                                advisorList.addAll(moreAdvisorsList);
                }
        }

}

private Double computeTotalSeniority(List<BuyingAgent> advisorList, long t) {
        Double totalSeniority = 0.0;
        for (BuyingAgent a : advisorList) {
                totalSeniority += calculateSeniority(a, t);
        }
        return totalSeniority;
}

private void blacklistAdvisors(List<BuyingAgent> unfavorableAdvisors) {
        if (m_advisorBlacklist == null)
                m_advisorBlacklist = new Vector<BuyingAgent>();
        m_advisorBlacklist.addAll(unfavorableAdvisors);
}

private Vector<BuyingAgent> removeUnfavorableAdvisors(
                List<BuyingAgent> advisorList, Double threshold) {
        Vector<BuyingAgent> removedAgents = null;
        for (BuyingAgent a : advisorList) {
                Double weightedAdvisorRating = getWeightedAdvisorRating(a);
                if (weightedAdvisorRating != null
                                && weightedAdvisorRating < threshold) {
                        if (removedAgents == null)
                                removedAgents = new Vector<BuyingAgent>();
                        removedAgents.add(a);
                }
        }
        setAdvisorList(removedAgents);
        return removedAgents;
}

private void setAdvisorList(List<BuyingAgent> unfavorableAdvisors) {
        if (unfavorableAdvisors != null) {
                List<BuyingAgent> currentAdvisorList = getAdvisors();
                Vector<BuyingAgent> newAdvisorList = new Vector<BuyingAgent>();
                for (BuyingAgent ba : currentAdvisorList) {
                        if (!unfavorableAdvisors.contains(ba))
                                newAdvisorList.add(ba);
                }
                setAdvisors(newAdvisorList);
        }
}
```

```
	private Double getWeightedAdvisorRating(BuyingAgent a) {
		Double weightedRating = null;
		if (m_advisor_ratings != null) {
			Vector<Rating> ratingVector = m_advisor_ratings.get(a);
			if (ratingVector != null)
				weightedRating = computeWeightedRating(ratingVector);
		}
		return weightedRating;
	}

	private Double computeWeightedRating(Vector<Rating> ratingVector) {
		Double weightedRating = null;
		Double totalRating = 0.0;
		Double totalSeniority = 0.0;
		int nRatings = 0;
		for (Rating ar : ratingVector) {
			if (ar instanceof SeniorityRating) {
				SeniorityRating sr = (SeniorityRating) ar;
				totalRating += sr.getRating() * sr.getRateeSeniority();
				totalSeniority += sr.getRateeSeniority();
				nRatings++;
			}

		}
		if (totalSeniority > 0)
			weightedRating = totalRating / totalSeniority;
		// if we have a rating but the seniority is zero (no transactions) then
		// we have a malicious advisor
		else if (totalSeniority == 0 && nRatings > 0)
			weightedRating = -1.0;
		return weightedRating;
	}

	void printAdvisorList(List<BuyingAgent> baList) {
		if (baList == null)
			System.out.println("null");
		else {
			for (BuyingAgent ba : baList) {
				System.out.print(ba.getId() + ", ");
			}
			System.out.print("\n");
		}

	}
}
```

Appendix B

Sample Experimental Results

As a sample of the results obtained during the experimental analysis, the following summary tables are presented for Experiment #1:

- Average Utility All Purchases;
- Average Utility First Ten Purchases;
- Good Vendors (%) All Purchases; and
- Good Vendor (%) First Ten Purchases.

Experiment #1: Balanced Honesty

Average Utility - All Purchases

Random Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	6.77	6.97	6.72	6.88	6.87	6.54	7.07	6.44	6.74	6.28	6.55
2	6.98	7.12	6.75	6.89	7.23	7.25	7.22	7.00	6.37	6.48	7.01
3	6.96	6.90	6.89	6.72	6.89	7.37	7.24	7.05	6.47	7.27	7.12
4	6.93	6.72	6.89	7.04	7.10	6.74	6.84	7.28	7.13	7.04	6.73
5	6.68	7.20	6.58	6.51	6.54	6.59	6.90	6.15	6.76	6.80	6.68
6	6.84	6.61	7.27	6.79	6.38	6.90	6.84	6.91	7.27	6.78	6.55
7	6.94	7.15	7.05	6.36	6.96	7.82	6.66	6.78	6.69	6.95	6.28
8	6.96	7.12	7.16	7.26	7.18	6.16	6.78	6.34	7.19	7.25	6.51
9	6.74	6.82	6.88	6.53	6.33	6.78	6.83	6.70	7.09	6.90	7.46
10	6.95	7.32	6.63	6.92	7.19	6.89	6.91	6.69	7.10	6.64	6.92
Total	6.88	7.00	6.88	6.79	6.87	6.90	6.93	6.73	6.88	6.84	6.78

Random Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	7.95	8.07	8.13	8.01	7.99	7.83	7.85	7.87	7.78	7.54	7.42
2	8.02	8.17	8.08	8.01	7.94	8.01	7.99	8.05	7.95	7.95	7.51
3	8.07	8.06	8.07	8.13	8.08	8.11	8.09	8.09	8.09	7.67	7.86
4	7.94	7.94	8.06	8.02	8.04	8.08	7.87	7.79	7.60	7.76	7.04
5	7.97	8.13	8.11	7.96	8.01	7.84	8.01	7.87	7.88	7.45	7.63
6	7.99	8.07	7.96	8.15	8.01	7.96	7.92	8.02	7.75	7.77	7.72
7	7.97	7.99	8.00	8.14	8.04	8.03	7.81	7.99	7.80	7.84	7.04
8	7.99	8.12	8.07	8.01	8.06	7.98	7.83	7.93	7.82	7.69	7.63
9	8.02	8.12	8.03	8.13	8.02	8.11	7.86	7.99	7.81	7.72	7.47
10	7.98	7.99	8.14	8.05	8.02	8.00	7.95	8.03	7.60	7.65	7.27
Total	7.99	8.07	8.06	8.06	8.02	7.99	7.92	7.96	7.81	7.70	7.46

Seniority Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	8.27	7.23	8.51	8.50	8.45	8.43	8.43	8.41	8.42	8.44	8.46
2	7.85	7.42	7.48	7.99	7.99	8.05	8.05	8.05	8.09	8.11	8.10
3	5.96	6.19	5.71	6.26	5.86	5.95	5.89	5.97	5.88	5.91	5.85
4	8.28	7.78	8.24	8.47	8.41	8.35	8.35	8.39	8.34	8.37	8.37
5	8.28	7.53	8.34	8.32	8.46	8.47	8.48	8.46	8.45	8.42	8.43
6	8.23	7.54	8.05	8.35	8.47	8.46	8.46	8.49	8.47	8.45	8.47
7	8.38	7.69	8.51	8.51	8.50	8.50	8.51	8.51	8.52	8.48	8.53
8	8.06	7.09	8.40	8.51	8.13	8.15	8.15	8.04	8.09	8.08	8.14
9	7.92	7.64	7.76	8.42	7.89	7.90	7.87	7.96	7.91	7.96	7.94
10	8.37	7.69	8.49	8.51	8.50	8.50	8.50	8.48	8.54	8.51	8.52
Total	7.97	7.40	7.95	8.18	8.06	8.08	8.07	8.08	8.07	8.07	8.08

Seniority Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	8.48	8.37	8.52	8.49	8.49	8.52	8.48	8.50	8.50	8.50	8.53
2	8.49	8.39	8.51	8.50	8.51	8.51	8.49	8.50	8.48	8.49	8.48
3	8.47	8.35	8.50	8.50	8.50	8.49	8.50	8.50	8.49	8.51	8.53
4	8.48	8.41	8.49	8.50	8.50	8.52	8.48	8.46	8.49	8.51	8.45
5	8.49	8.45	8.52	8.48	8.51	8.50	8.51	8.53	8.50	8.48	8.46
6	8.37	8.36	8.49	8.50	8.40	8.26	8.18	8.18	8.44	8.31	8.42
7	8.46	8.33	8.50	8.48	8.49	8.49	8.49	8.48	8.43	8.49	8.48
8	8.49	8.42	8.50	8.49	8.51	8.51	8.50	8.48	8.53	8.51	8.47
9	8.45	8.28	8.50	8.49	8.49	8.47	8.48	8.46	8.44	8.47	8.44
10	8.49	8.43	8.51	8.49	8.48	8.52	8.50	8.49	8.50	8.51	8.46
Total	8.47	8.38	8.50	8.49	8.49	8.48	8.46	8.46	8.48	8.48	8.47

Experiment #1: Balanced Honesty

Average Utility - First Ten Purchases

Random Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	6.30	5.31	5.42	6.85	6.63	6.24	7.23	5.90	6.02	6.82	6.32
2	6.45	5.25	5.54	6.12	6.80	7.38	6.89	6.34	6.23	6.58	7.18
3	6.36	5.43	6.04	5.99	6.19	6.87	6.62	6.63	5.72	7.01	6.96
4	6.53	4.96	6.03	6.33	6.41	6.57	7.06	7.13	6.82	7.45	6.43
5	6.09	5.82	5.36	5.70	5.71	5.94	6.33	6.36	6.18	7.35	6.19
6	6.32	5.40	5.84	6.40	5.53	6.44	6.36	6.52	7.39	6.48	6.74
7	6.42	4.39	5.96	5.95	6.64	7.60	6.53	6.72	6.39	6.89	6.65
8	6.30	5.13	6.20	6.53	6.51	5.92	6.06	6.74	6.61	6.94	6.29
9	6.31	5.92	5.24	5.76	5.90	6.36	6.07	6.43	6.79	7.19	7.31
10	6.55	5.30	5.94	6.71	6.78	6.78	6.58	6.36	7.11	6.69	7.20
Total	6.36	5.29	5.76	6.23	6.31	6.61	6.57	6.51	6.53	6.94	6.73

Random Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	6.73	6.42	6.72	7.20	6.62	6.78	6.73	6.51	7.05	6.48	6.73
2	6.57	5.10	6.42	6.46	5.91	6.64	6.27	7.44	6.96	7.68	6.68
3	6.77	5.86	6.53	6.24	5.91	7.01	7.24	7.04	7.72	6.58	7.39
4	6.65	5.31	6.29	6.27	7.28	6.61	6.71	6.58	7.23	7.18	6.85
5	6.40	5.46	6.61	5.85	5.86	6.31	7.10	6.81	6.90	6.17	6.83
6	6.33	5.40	5.85	6.15	5.90	6.83	7.07	5.93	6.35	6.89	6.86
7	6.42	4.63	6.11	6.37	6.63	6.60	6.81	7.07	6.00	7.42	6.41
8	6.58	6.10	5.99	6.43	6.77	6.89	6.77	6.83	6.60	6.69	6.68
9	6.81	5.80	6.26	6.96	6.78	6.83	6.87	6.94	7.71	7.23	6.80
10	6.53	4.43	6.37	6.67	6.58	6.83	6.73	7.32	6.36	7.04	6.64
Total	6.58	5.46	6.32	6.47	6.42	6.73	6.83	6.85	6.89	6.93	6.79

Seniority Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	8.10	5.37	7.94	8.46	8.66	8.36	8.46	8.28	8.13	8.38	8.51
2	7.60	4.82	7.18	7.94	7.95	8.21	7.62	8.27	7.74	7.94	8.09
3	5.83	5.74	5.73	6.31	5.71	5.69	6.08	5.56	5.54	6.17	5.76
4	8.08	5.35	8.12	8.37	8.51	8.20	8.26	8.17	8.33	8.37	8.40
5	8.13	4.91	8.04	8.40	8.37	8.61	8.61	8.48	8.62	8.33	8.36
6	7.99	4.92	7.17	8.19	8.60	8.44	8.49	8.45	8.34	8.49	8.41
7	8.20	4.51	8.54	8.60	8.54	8.51	8.41	8.71	8.46	8.42	8.57
8	8.04	5.74	8.42	8.49	8.15	8.31	8.22	7.96	8.49	8.10	8.18
9	7.68	5.35	7.61	8.40	8.05	7.82	7.99	7.88	8.06	7.43	7.72
10	8.20	5.15	8.48	8.55	8.54	8.56	8.49	8.46	8.60	8.48	8.35
Total	7.79	5.17	7.72	8.18	8.11	8.07	8.06	8.02	8.03	8.01	8.03

Seniority Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	8.44	7.85	8.81	8.24	8.44	8.58	8.46	8.49	8.44	8.48	8.51
2	8.33	6.42	8.46	8.49	8.76	8.46	8.59	8.52	8.44	8.59	8.40
3	8.49	8.26	8.48	8.59	8.39	8.69	8.55	8.53	8.41	8.71	8.30
4	8.47	8.39	8.41	8.36	8.73	8.61	8.36	8.48	8.61	8.50	8.29
5	8.41	7.24	8.54	8.48	8.48	8.48	8.45	8.60	8.60	8.40	8.71
6	8.32	6.58	8.55	8.36	8.55	8.37	8.63	8.46	8.58	8.33	8.45
7	8.03	3.96	8.43	8.44	8.43	8.43	8.52	8.42	8.39	8.49	8.46
8	8.47	8.07	8.41	8.51	8.43	8.46	8.43	8.46	8.61	8.60	8.63
9	8.35	7.41	8.32	8.54	8.50	8.44	8.31	8.34	8.53	8.61	8.36
10	8.42	7.57	8.47	8.43	8.63	8.43	8.35	8.65	8.56	8.55	8.50
Total	8.37	7.20	8.49	8.44	8.53	8.49	8.47	8.50	8.52	8.53	8.46

Experiment #1: Balanced Honesty

Good Vendors (%) - All Purchases

Random Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	64.72	68.98	63.80	67.18	65.96	59.91	71.62	56.80	64.14	54.67	60.45
2	69.47	71.90	64.95	67.49	73.79	75.38	73.54	70.18	59.06	58.73	70.11
3	68.85	65.25	66.14	66.00	68.34	76.54	73.36	73.58	61.97	74.45	71.43
4	67.74	62.99	65.88	70.49	72.13	63.80	66.18	73.91	72.27	69.36	64.24
5	64.08	75.02	61.04	59.94	61.34	61.77	68.90	55.27	65.63	65.34	65.43
6	66.39	61.50	75.82	65.88	57.21	67.17	65.78	67.87	74.25	65.57	60.87
7	68.81	73.73	70.63	57.66	68.84	86.10	63.35	65.25	63.73	68.68	55.05
8	68.12	70.82	71.80	73.71	72.76	52.55	64.31	58.45	73.33	74.33	58.13
9	64.22	65.19	67.49	58.64	57.71	65.96	65.14	63.88	71.12	67.47	77.57
10	68.67	76.38	63.73	67.51	73.25	67.11	66.66	63.60	71.67	61.48	66.80
Total	67.12	69.30	67.13	65.45	67.13	67.63	67.89	64.88	67.71	66.00	65.03

Random Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	88.69	91.05	92.54	89.61	89.71	86.07	86.71	87.24	84.86	80.52	77.23
2	90.00	92.97	91.23	90.04	88.63	89.54	88.68	90.11	87.82	88.75	80.13
3	90.83	90.33	91.52	91.90	90.82	91.91	91.60	91.49	89.80	82.95	86.60
4	88.46	89.04	90.55	89.91	90.48	91.09	86.35	86.07	82.48	83.99	71.11
5	89.06	92.36	91.54	88.73	90.12	86.74	90.09	86.86	87.46	76.92	81.84
6	89.43	91.11	88.63	93.01	90.18	88.89	88.03	89.27	84.97	84.73	81.98
7	88.98	89.69	89.34	92.32	90.31	89.99	85.50	88.96	85.24	85.82	72.14
8	89.56	91.53	91.26	90.75	90.64	89.19	87.01	88.23	86.38	83.54	82.71
9	89.99	91.53	90.59	92.04	90.45	91.88	86.82	90.06	84.89	84.51	79.38
10	89.32	89.42	92.68	90.77	89.96	89.23	89.33	90.46	81.08	82.35	77.63
Total	89.44	90.93	90.99	90.91	90.13	89.45	88.01	88.87	85.49	83.41	79.08

Seniority Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	95.11	73.65	99.70	100.00	99.01	98.41	98.25	97.92	98.43	98.53	98.94
2	85.48	78.55	77.85	89.09	88.21	88.94	89.51	88.93	88.89	89.19	90.11
3	48.74	52.13	42.24	53.02	48.18	49.53	49.00	49.63	48.77	47.00	48.27
4	94.88	84.47	94.75	98.89	96.94	96.21	96.18	96.57	95.96	96.23	96.94
5	95.48	79.15	97.11	97.17	98.64	99.39	99.06	98.89	98.70	99.33	99.06
6	94.31	80.24	89.82	96.15	99.50	99.62	99.65	99.70	99.73	99.66	99.33
7	97.56	84.25	99.94	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8	89.50	71.88	97.88	99.80	89.75	90.20	90.30	87.98	88.29	87.74	88.56
9	86.29	81.97	81.01	97.98	85.27	85.92	85.49	86.97	85.70	86.56	86.55
10	97.38	83.08	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Total	88.59	77.31	88.04	93.21	90.53	90.83	90.74	90.67	90.43	90.44	90.78

Seniority Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	99.58	97.50	99.92	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
2	99.68	98.06	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
3	99.31	96.33	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4	99.74	98.62	99.95	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5	99.83	99.15	99.91	99.95	100.00	100.00	99.95	100.00	100.00	100.00	100.00
6	97.71	95.86	99.85	99.95	98.52	96.36	94.97	95.27	98.83	97.70	98.40
7	99.38	96.44	100.00	100.00	100.00	100.00	100.00	100.00	99.37	100.00	100.00
8	99.75	98.44	99.94	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9	98.78	94.62	100.00	100.00	99.56	99.44	99.33	99.19	99.02	99.66	99.47
10	99.74	98.48	99.94	100.00	100.00	100.00	100.00	100.00	100.00	99.87	100.00
Total	99.35	97.35	99.95	99.99	99.81	99.58	99.42	99.44	99.72	99.72	99.79

Experiment #1: Balanced Honesty

Good Vendors (%) - First Ten Purchases

Random Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1.00	54.60	31.25	44.44	65.43	55.42	54.43	72.50	45.12	53.09	63.86	55.70
2.00	58.09	26.39	42.31	51.85	59.26	75.00	70.37	54.43	55.00	64.56	76.83
3.00	56.77	39.13	50.00	47.56	51.90	64.63	63.41	61.45	51.28	67.09	68.29
4.00	57.93	22.97	47.50	54.32	59.26	53.66	70.37	69.62	60.98	75.61	62.03
5.00	53.12	41.10	44.44	40.24	47.50	53.09	59.26	59.52	57.83	68.83	58.75
6.00	55.00	34.72	50.63	52.50	42.17	58.75	56.25	57.32	72.50	59.76	63.41
7.00	57.27	26.56	49.38	47.62	60.00	81.48	55.56	60.24	57.32	67.07	60.98
8.00	54.64	30.99	51.76	61.25	59.26	44.19	50.00	59.26	65.00	68.75	54.22
9.00	55.16	40.68	37.97	47.44	50.00	60.00	47.62	58.23	67.50	64.56	73.49
10.00	59.39	38.36	46.43	62.65	61.73	62.96	60.49	58.14	69.88	63.86	67.50
Total	56.20	33.14	46.54	53.08	54.65	60.78	60.52	58.31	61.06	66.38	64.16

Random Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1.00	62.41	52.05	61.45	65.06	63.10	55.42	64.56	65.38	70.11	59.26	66.27
2.00	60.99	25.00	61.18	53.66	46.91	66.67	54.22	75.31	71.08	84.15	67.50
3.00	64.08	40.58	60.00	52.50	50.00	70.00	71.25	70.73	73.49	68.75	79.52
4.00	61.74	41.54	54.22	54.76	66.25	64.63	60.98	62.35	75.00	70.73	63.41
5.00	57.78	38.57	60.00	50.00	48.81	53.57	71.60	61.63	67.50	55.95	68.42
6.00	58.01	38.89	50.00	57.32	42.86	68.75	72.84	49.40	62.50	69.51	66.67
7.00	59.43	28.77	50.00	57.32	62.65	68.29	63.29	68.29	54.32	77.22	61.54
8.00	61.64	49.30	47.62	60.98	67.07	62.50	63.86	68.75	61.73	67.50	66.25
9.00	63.78	43.02	52.44	68.67	61.45	65.85	67.07	66.25	75.90	72.50	65.82
10.00	60.34	24.29	53.01	62.65	60.24	61.25	63.41	76.47	57.65	69.14	70.00
Total	61.03	38.28	54.99	58.36	56.90	63.64	65.27	66.42	66.95	69.42	67.58

Seniority Base Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1.00	92.67	34.33	92.86	100.00	100.00	98.77	100.00	97.56	95.12	97.56	100.00
2.00	78.32	25.35	69.88	81.71	85.00	88.75	84.15	88.46	82.14	80.25	92.21
3.00	47.61	40.35	44.44	56.25	46.34	51.85	54.22	41.18	42.68	49.40	47.56
4.00	90.30	35.00	88.46	96.30	97.50	95.18	95.06	95.18	96.30	96.39	92.86
5.00	92.63	27.27	90.00	97.53	100.00	100.00	100.00	100.00	98.81	100.00	100.00
6.00	89.14	22.54	72.09	92.59	98.80	98.77	100.00	100.00	98.78	100.00	100.00
7.00	93.47	20.90	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	87.47	42.03	96.30	100.00	92.41	93.83	93.83	84.71	95.00	86.59	83.13
9.00	82.35	35.38	82.93	98.78	85.37	83.13	81.25	90.00	90.36	81.25	85.37
10.00	94.44	37.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Total	84.89	31.88	83.62	92.45	90.55	91.06	90.72	89.60	89.91	89.13	90.07

Seniority Model											
Run	All	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1.00	98.36	81.69	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
2.00	94.79	43.24	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
3.00	99.39	94.19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4.00	99.39	94.05	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.00	97.91	77.63	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	96.33	56.06	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	91.94	12.16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	99.25	91.04	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	97.62	73.97	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10.00	98.38	82.43	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Total	97.34	71.28	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00