ICT SHOK FI Programme: Deliverable 6.3.2.7

Research report: Reputation system simulation results

(The attached Master Thesis consitutes the deliverable)

Yiyun Shen
December 10, 2009
Department of Computer Science
University of Helsinki
Helsinki Institute for Information Technology (HIIT)

Date of acceptance Grade

Instructor

The Simulation of Reputation Systems in Widget Sharing Communities: A Comparison Study

Yiyun Shen

Helsinki November 30, 2009

M.Sc. Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Yiyun Shen

Työn nimi — Arbetets titel — Title

The Simulation of Reputation Systems in Widget Sharing Communities: A Comparison Study

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
M.Sc. Thesis	November 30, 2009	69 pages + 0 appendices

Tiivistelmä — Referat — Abstract

Nowadays, widgets, small interactive applications for displaying and updating local data or online data, are frequently shared between developers and users in many online communities and websites. However, users often have no previous experience with the widgets they are going to download, or they have no interaction history with the developers who have submitted the widgets. The main risks for the users include (i) the downloaded widgets cannot provide the functions or utilities as expected, and (ii) these widgets may contain computer virus. These risks can be avoided by employing a so-called reputation system, the computational mechanism that collects and aggregates feedback information from users and distributes reputation information to other users. To ensure that a reputation system works as expected before it is deployed to a real system, the system should be verified using multi-agent simulation.

This thesis compares and evaluates six reputation systems within a simulated widget sharing community. The goal is to find out the strengths and weakness of these systems when applying in a specific widget sharing domain. For the purpose of this comparison study, a custom scenario was proposed and a simulation environment was formed using statistical models that are determined by real data. The effectiveness of different reputation systems were evaluated and compared by simulation. The experiment results indicate to which degree the evaluated systems can stand the attacks from different kinds of misbehaving developers and users, as well as how effective the systems are in providing reliable reputation information of widgets to users and supporting high downloads of honest widgets.

ACM Computing Classification System (CCS):
D.4.7 Organization and design - Distributed systems,
H.5.2 Information interfaces and presentation - User interfaces,
H.3.5 Information storage and retrieval - On-line information systems,
H.3.4 Information storage and retrieval - Systems and software

Avainsanat — Nyckelord — Keywords
simulation, reputation system, widget sharing

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Acknowledgments

Contents

1 Introduction

Nowadays, an increasing number of websites, such as SourceForge.net[1] and various widget sharing websites, allow software developers to share code with other people on the Internet. Downloading software from these websites contains a risk as the software may not work as expected or it may be malicious, e.g., contain a computer virus. Up to now, few of these websites provide decision support mechanisms for determining the nature of widgets beforehand.

This situation can be improved by employing a so-called reputation system, a computational mechanism that collects and aggregates feedback information from users and distributes reputation information to other users. Due to the potential risks involved in download decisions, it should be ensured that a reputation system works as expected before it is deployed to a real system.

A possible way to verify that the system works properly is to use multi-agent simulations. Multi-agent simulations have been previously used to study the effectiveness and efficiency of reputation systems in other domains, such as e-commerce [Del03, Nur07], peer-to-peer systems [GJA03, YSS04], or mobile ad-hoc networking [BLB02]. However, the widget sharing domain differs from these domains in two ways: (i) the risks involved in decision making are not directly measurable, because widgets, as transaction objects, have no monetary value and the harm of malicious widgets is not predictable; and (ii) on widget sharing sites, "sellers" (developers) provide widgets and "buyers" (users) act as consumers of the virtual products, thus, there are no direct interactions between users and developers.

This thesis aims to analyze the effectiveness of reputation systems in a simulated widget sharing community. In this thesis, a custom scenario is proposed to represent a real widget sharing website. Within this scenario, a reputation system is regarded as effective, if it can distinguish the good widgets from the malicious ones, i.e., reward the good widgets and their developers with more download transactions and penalize the malicious widgets and their developers with less downloads. The effectiveness of different reputation systems are examined and compared by simulation. The idea is to evaluate to which degree the systems can stand the attacks from different kinds of misbehaving developers and users. To ensure that the experiment results are realistic, we refer to empirical data collected from a real widget sharing website for our simulation setup. In the experiment results, the author illustrates the maximum

[1] `http://sourceforge.net/` [Retrieved: 2009-09-04]

amount of dishonest developers or users at which the reputation systems can resist.

The rest of this thesis is organized as follows. Section 2 draws research questions of this work and provides background information on the concepts used throughout the thesis. Section 3 surveys the state of the art of reputation systems and Section 4 reviews different multi-agent simulation techniques. Section 5 describes our simulation setup. Section 6 describes the experiments that are conducted to evaluate various reputation systems and discusses the strengths and weaknesses of the reputation systems. Section 7 summaries the contributions of this thesis and discusses limitations of this work.

2 Reputation Management in Widget Sharing Communities

This thesis work seeks to improve trust between users and developers in widget sharing communities in accordance with reliable reputation information about widgets and developers. This research objective leads us to consider two problems that will be studied in this thesis:

1. Build a simulation environment to model a real world widget sharing community.
2. Evaluate the effectiveness of different reputation systems within the simulated widget sharing community.

This section provides background information concerning *widgets and widget sharing websites* (Section 2.1), *trust and reputation* (Section 2.2), *reputation system* (Section 2.3) and *multi-agent simulation* (Section 2.4), as well as the connections between these concepts.

2.1 Widget and Widget Sharing Websites

The notion *widget*, a combination of *window* and *gadget*, was born with the invention of the graphical user interface (GUI) and it was first used to name user interface elements during Project Athena in 1988 [AM90]. Originally, *widget* was used to refer to an element of GUI that displays changeable information arrangements, such as a window or a text box. Following years of development, *desktop widget*, a closely related concept, has emerged to describe small specialized GUI applications that provide visual information and easy access to computing functions like clocks, calendars, calculators and small games. Another related concept is the *web widget*, a user interface element that can be inserted into webpages to show web contents, such as news headlines, weather forecasts or Wikipedia articles.

In this thesis, a *widget* is referred as an interactive application for displaying and/or updating local data or data on the Web, packaged to allow a download and installation on a user's machine or mobile device [CP09]. This definition covers both desktop and web widgets, but not GUI widgets. Widgets can be classified into various categories, e.g., news, weather sites, currency or unit converters, CPU gauges, dictionaries and games [BNF+08].

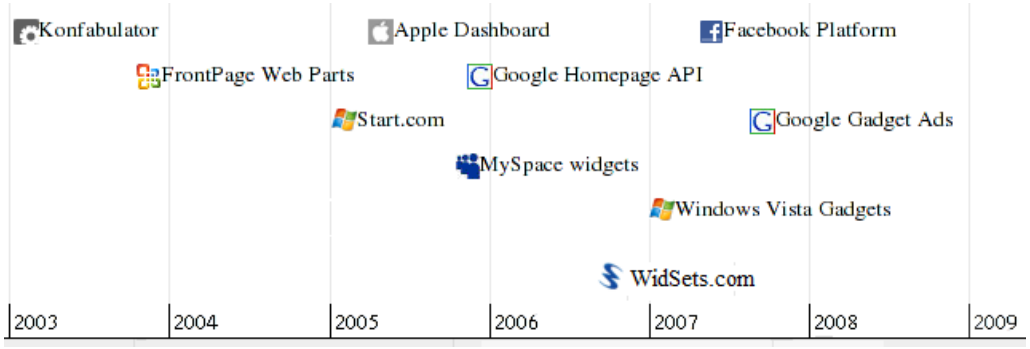

Figure 1: A timeline modified from Niall Kennedy's widget timeline[9]shows the launch time of several widget sectors starting from 2003. The years 2005 and 2007 saw a bloom of widget sectors.

Widgets, especially desktop widgets, are typically hosted by a widget engine. Examples of widget engines include a JavaScript runtime engine called Konfabulator and Dashboard, a platform that allows users to place widgets on the Mac OS X desktop. Many browsers and widget engines are linked with online widget sharing websites that allow users to download new widgets. Examples of widget sharing websites or communities include Dashboard Widgets[2], Yahoo! Widgets[3], Windows Live[4], Google Gadgets[5], Widgipedia[6], Widgetbox[7] and Nokia WidSets[8]. Figure 1 shows the history of widget platforms and online widget sharing websites that have been launched since 2003.

Figure 2 illustrates the main page of a widget sharing website, Yahoo! Widgets, which acts as a platform to connect people who want to share codes with others (developers) and those who like to download and use widgets (users). Links that lead widget users to *Find Widgets* and guide developers to *Create Widgets* are shown in the navigation bar on the top of the webpage. The main body of the webpage contains various parts, such as *Widgets Exhibition*, *Search Bar*, *Lists of Widgets* and *Widget Categories*, which can help users to find widgets that meet their potential

[2]`http://www.apple.com/downloads/dashboard` [Retrieved: 2009-09-01]
[3]`http://widgets.yahoo.com` [Retrieved: 2009-09-01]
[4]`http://home.live.com` [Retrieved: 2009-09-01]
[5]`http://www.google.com/webmasters/gadgets` [Retrieved: 2009-09-01]
[6]`http://www.widgipedia.com` [Retrieved: 2009-09-01]
[7]`http://www.widgetbox.com` [Retrieved: 2009-09-01]
[8]`http://www.widsets.com` [Retrieved: 2009-04-20]
[9]`http://www.niallkennedy.com/blog/timelines/widgets/` [Retrieved: 2009-08-27].
[10]`http://widgets.yahoo.com/` [Retrieved: 2009-11-18].
[11]`http://widgets.yahoo.com/tools/` [Retrieved: 2009-11-18].

Figure 2: Screenshot of the main page of Yahoo! Widgets[10], a widget sharing website. The webpage provides a navigation bar with links that help widget users to *Find Widgets* and guide developers to *Create Widgets*. The main body of the webpage contains *Widgets Exhibition*, *Search Bar*, *Lists of Widgets* and *Widget Categories*, to improve the accessibility to widgets of user interests.

Figure 3: Screenshot of a webpage[11] providing tutorials and tools for widget developers. This webpage also contains a link that guide developers to submit their widgets to the website.

needs. The *Widgets Exhibition* shows generic information about new widgets, e.g., widget's name, developer, rating, picture, description and a *Get it!* button for user to download the widget. Users can also search for widgets with key words in *Search Bar*; reach, for instance, new widgets via *Lists of Widgets*; or select a certain kind of widgets within *Widget Categories*. For developers, by clicking the *Create Widgets* link, both tutorials and tools needed for building and submitting widgets are available in the website; see Figure 3. Figure 4 demonstrates a detailed view of *Yahoo! Weather*, an example of the existing widgets, shown in Yahoo! Widgets.

2.2 Trust and Reputation

Trust and reputation have received considerable attention within various scientific disciplines, such as psychology, sociology, philosophy and economy; see [MM02, SS05]. Thus, it is not surprising that various definitions of reputation have been given and none of them has received widespread acceptance [MMH02]. Moreover, the term reputation is often mixed up with another concept, i.e., trust [ARH00].

Trust is defined as a subjective probability assessed by an agent that another agent or group of agents will perform a particular action or decision, before the agent can monitor such an action [Gam88]. The assessment is based on the risks, benefits and reputations of the agents involved in the situation. *Reputation*, on the other hand, is defined as the collected and processed information about an individual's past behavior as experienced by others [SVB06]. Accordingly, trust is a subjective judgment that is based on various factors or evidence, but reputation is a collective measure of trustworthiness or reliability based on the referrals or ratings from members in a community [JIB07]. In decision making, personal experience, i.e., trust, usually carries more weight than second-hand referrals, i.e., reputation. However, if direct experiences are lacking, especially when a trust relationship is initialized between both parties, decision making has to be based on referrals from others [RK05].

2.3 Reputation System

A *reputation system* is a computational system that collects, distributes, and aggregates feedback information about individuals' past behavior [RKZF00]. Reputation systems are applied to provide an incentive for honest behavior and to help people

[12] `http://widgets.yahoo.com/widgets/yahoo-weather` [Retrieved: 2009-11-18].

Figure 4: A detailed view of the widget *Yahoo! Weather*[12]with information of the widget and feedback elements, such as average rating, number of ratings, number of downloads, and reviews.

make decisions about who to trust. For example, on Amazon.com, consumers can rate the products from 1 (least satisfied) to 5 (most satisfied) and the average of ratings a product received is regarded as the reputation value of the product. Some widget sharing websites, like Yahoo! Widgets, have adopted this kind of reputation systems as well. In Yahoo! Widgets, users are informed about whether a widget is trustworthy or not in terms of average rating in a five-star scale and number of ratings; see Figure 4. This adoption of reputation systems in widget sharing communities can be regarded as a possible solution to provide users the reputation information about widgets. However, Hu et al. [HPZ06] argue that the average score of a product in Amazon.com may mislead the consumers if the reputation of the product fails to reveal the product's true quality. There also remains a question mark about whether the reputation systems applied in widget sharing communities can effectively guide users to download trustworthy widgets or not.

To this end, we have to review different kinds of reputation systems that can be applied in widget sharing communities (Section 3) and compare their effectiveness in providing users reliable reputation information about widgets and encouraging the users to download good widgets. However, it is impossible to simply replace one reputation system with another one and compare several different reputation systems within one real website, because the tests can cost a large amount of workload on the website side and cause confusion due to inconsistent reputation information offered to the user side. To evaluate and compare the effectiveness of various reputation systems in a widget sharing scenario requires a reasonable solution. Thus, we refer to computer simulation.

2.4 Using Multi-Agent Simulation for Comparison Study

Computer simulation attempts to model a real-life or a hypothetical situation in a computational way so that it can be studied and understood [RRH00]. Computer simulation is a powerful tool that allows us to stay with a problem and choose the best alternative by providing a way in which various designs, plans and/or policies can be evaluated without having to experiment on a real system, which may be prohibitively costly, time-consuming, or simply impractical to do. For example, reputation systems should be evaluated using simulation before being put into actual use, because to examine reputation systems in a real system causes potential risks, e.g., possible monetary loss, to participants, if they make their decisions based on incorrect information generated by reputation systems.

Basic computer simulation alone, however, cannot model all the real world systems that are needed for the study of reputation systems. Demands from certain aspects, like modeling of intelligent human behavior in widget sharing communities, are beyond the control of standard simulations. To this end, a multi-agent mechanism provides a way to model the human behavior by abstracting the individuals into computer agents that are assigned with pre-defined behavior models.

Multi-agent simulation is a multi-agent based model concept that combines a multi-agent mechanism (Section 4.1) with a simulated environment in virtual time. In multi-agent simulation, active entities are modeled as agents who live in an environment consisting of other agents, and environmental or physical objects, such as resources, are modeled as entities. The key point of multi-agent simulation is interactions between agents who autonomously make decisions according to their pre-defined behavior models [DF94].

In our work, we consider multi-agent simulation to provide the possibility of directly representing individuals (as agents) and their behavior during interactions in a computerized form. For instance, individuals are directly represented in the form of agents, i.e., users and developers in a widget sharing community, characterized by different behavior models. An agent can make its own decision when interacting with other agents, and the behavior model of the agent belongs affects its decision making process; see Section 5.2. Thanks to the flexibility and capability for the integration of multi-agent simulation, we are able to set up our simulation using:

1. differential equations, i.e., metrics in the reputation systems (Section 3.4);
2. quantitative variables, i.e., simulation parameters (Section 5.4); and
3. qualitative parameters, i.e., individual behavior (Section 6.1.1).

These variables integrated in the multi-agent simulation enable the comparison of effectiveness of different reputation systems. In this thesis, a reputation system employed in a widget sharing community is regarded as effective, if it can distinguish honest widgets from malicious ones and encourage users to download trustworthy widgets. We return to this topic in Sections 5 and 6 with more details about the simulation setup and the evaluation of reputation systems.

3 Reputation Systems

In order to provide users reliable reputation information about widgets, reputation systems are considered to be deployed in widget sharing communities. Section 3.1 describes dimensions for classifying reputation systems, and Section 3.2 covers the notation used in metrics in reputation systems. Section 3.3 reviews reputation systems that have been presented in previous work and illustrates several quantitative metrics, which act as the foundation of this comparison study of reputation systems. Section 3.4 summarizes the characteristics of the reviewed systems.

3.1 Classification Dimensions

Research of reputation systems under different conditions makes it a bit difficult to distinguish them using a single dimension. Considering the characteristics of reputation systems, we adopt two classification categories: information sources and visibility types, from [SS05].

Information Sources The information that a reputation system uses to estimate the reputation of an agent can originate from different sources, such as direct experiences, witness information, and information related to the sociological features of agents' behavior [SS05].

Direct experiences are based on agents' direct interactions with their partners, and are considered the most relevant and reliable information source for a reputation system.

Witness information, also known as word-of-mouth or indirect information, is the information collected from other members of the community. For an agent, the information can come from its direct experiences or from others. Witness information can be replaced with direct experience, if the latter is reliable enough for a reputation model. It should also be taken into consideration that witnesses may manipulate or hide information to their own benefit.

Sociological information is based on the social relations between agents and the social status of these agents. The social relations between agents in a multi-agent system represent the complex relations between their human counterparts in real world situations. For instance, in a real community, an individual can play one or more roles, e.g., a buyer and/or a seller, in the community. Additionally, each

one can establish different kinds of relations with others, e.g., dependence, trade, competition or collaboration. The roles the individual plays and the relations (s)he has in the community can affect his or her behavior and the interaction with other individuals.

Visibility Types This classification dimension refers to how an agent's reputation is visible to all agents within a community. A reputation system is a *global* system if an agent's reputation is regarded as a global property accessible to all agents in the community and the agent has only one single reputation value. Alternatively, a reputation system is regarded as a *local* system, if an agent's reputation is viewed as a property evaluated by each agent and an agent has different reputation values depending on the requesting agent who queries the reputation value.

Within a global/centralized reputation system, the reputation value of an agent is calculated from the opinions of agents who interacted with the agent in the past. This reputation value is accessible to all members of the community and updated whenever a member submits a new evaluation for an agent. Local reputation systems, also called personalized reputation systems, provide different reputation values for different groups of agents [MMH02]. Each agent assigns a personalized reputation value to each member of the community according to its direct experiences, witness information or other sociological information. In other words, an agent's personalized reputation is more about its reputation from another agent's point of view.

3.2 Common Notation for Metrics in Reputation Systems

The classification dimensions discussed in the previous section provide the possibility to analyze reputation systems in a qualitative way. However, the study of reputation systems also requires quantitative metrics that can benefit further evaluation of reputation systems using simulation. To enable the abstraction and comparison of the core metrics in different reputation systems, the notation presented by Schlosser et al. [SVB06] is adopted. This section covers the basic notation used throughout the survey of reputation systems in Section 3.3.

The set of agents is denoted using A, and the context of a transaction is written as C. $T = \{0, 1, ..., t_{now}\}$ is the set of times. E denotes the set of encounters between different agents that have happened till now. An encounter consisting of information

about the peers in the context of the transaction is written as:

$$E = \{(a, b, c) \in A \times A \times C | a \neq b\}. \tag{1}$$

A rating denoted as $\rho(a, e)$ represents a mapping between a target agent $a \in A$ and an encounter $e \in E$ to the set of possible ratings Q:

$$\rho(a, e) : A \times E \rightarrow Q. \tag{2}$$

The set of ratings Q has various shapes, e.g., $Q_{eBay} = \{-1, 0, 1\}$ or $Q_i = [0, 1]$.

The subset of all encounters in which an agent $a \in A$ has completed a transaction and received a rating is

$$E_a := \{e \in E | \, (e = (a, \cdot, \cdot) \vee e = (\cdot, a, \cdot))\}. \tag{3}$$

The time-sorted list of E_a is defined as $\overline{E_a}$.

The set of all encounters between agent $a \in A$ and agent $b \in A$ with a valid rating for the agent a is

$$E_{a,b} := \{e \in E_a | \, (e = (a, b, \cdot) \vee e = (b, a, \cdot))\}. \tag{4}$$

An encounter between a and b at a given time t is written as $e^t_{a,b} \in E_{a,b}$. The operator # denotes the size of a set of a list.

The reputation of agent $a \in A$ is a mapping between the agent a and time $t \in T$. The most recent reputation value of the agent a is shown as $r(a) := r(a, t_{now})$.

Table 1 summaries the notation that allows the abstraction of specific metrics from different kinds of reputation systems that are studied in our simulations. More specific notation is represented when it is encountered the first time.

3.3 Survey of Reputation Systems

This section reviews previous work on reputation systems from both qualitative and quantitative perspectives. The reputation systems are grouped according to their different ways of computing reputation, and the emphasis is on the operative mechanisms of these systems. Different kinds of systems are described using the notation represented in Section 3.2, excluding implementation related aspects, such

Table 1: A summary of the notation that is used in the metrics in reputation systems.

Symbol	Description
A	The set of agents
C	The context of a transaction
T	The set of times $(t_0, t_1, ..., t_{now})$
E	The set of encounters
Q	The set of ratings
$\rho(a, e)$	A rating for a mapping between a target agent a and an encounter e to the set of possible ratings Q
E_a	The subset of all encounters E in which an agent a has completed a transaction and received a rating
$\overline{E_a}$	The time-sorted list of E_a
$E_{a,b}$	The set of all encounters between agent a and agent b with a valid rating
$e^t_{a,b}$	An encounter between a and b at a given time t
$\#$	The size of a set of a list
$r(a)$	The most recent reputation value of agent a

as the flow of information or the location of processing and storage of data. To describe them in a simple way, the systems will be analyzed excluding the value of a transaction.

3.3.1 Accumulative Systems

The reputation systems that calculate the reputation of an agent as the sum of all given ratings are called *accumulative systems*. One of the representatives of the accumulative systems employed in online marketplaces is the eBay system, which informs buyers about whether potential trading partners are trustworthy or not, and aims at making chiseling and cheating rare. The reputation mechanism used in eBay[13] is based on ratings given by both trading parties (i.e., buyers and sellers) after a transaction has been completed. The possible ratings are 1 (positive), 0 (neutral) and -1 (negative), and the overall reputation value of an agent $a \in A$ is

$$r(a) = \sum_{e \in E_a} \rho(a, e), \tag{5}$$

which is a sum of all ratings given to the agent [SVB06].

The eBay system uses the reputation of a user as a single value, which is collected and computed from the available feedback information [SS05]. Witness information coming from other agents who previously interacted with the target agent is the main information source to form the reputation of the agent.

In accumulative systems, an honest agent can benefit from its good behaviors in the way that the more often it behaves well, the higher reputation it earns. On the other hand, dishonest agents may give their partners some false or biased ratings, to which the accumulative systems are vulnerable [Nur07]. To overcome the negative effects caused by false or biased information, the reputation value should be calculated using a large enough number of opinions or ratings. In this way, the accumulative systems can ignore some of the bad ratings that an agent has received, if the agent makes enough good transactions in the future. Another disadvantage is that the accumulative systems allow an agent to behave badly in a certain amount of transactions and still to improve its overall reputation, if the fraction of bad ratings gained from bad behaviors is small enough.

[13] `http:\\www.eBay.com` [Retrieved 2009-09-04]

3.3.2 Average Systems

In average systems, the reputation for an agent is computed as the average of all ratings that the agent has received. The average value is regarded as the global reputation of the agent. Agents can use the set of ratings, i.e., $\{-1, 0, 1\}$, to rate their interaction partners. The reputation of an agent $a \in A$ is calculated using

$$r(a) = \frac{\sum_{e \in E_a} \rho(a, e)}{\#(E_a)}. \tag{6}$$

In average systems, agents are assumed to behave in the same way for most of their lifetime. As unusual ratings are given little weight in the computation, they only have little effect on the final reputation. This weakness might be used by some malicious agents to intentionally issue bad transactions.

One example of average systems is the Jurca and Faltings system [JF03]. In this system, a group of broker agents, also called R-agents, buy and aggregate reports from other agents and sell back reputation information to agents when they need it. Each R-agent collects and aggregates reputation reports in a centralized way, though the agents are distributed in the reputation system. Reputation reports are limited to the values 0 and 1 so that the value 0 represents cheating agents and value 1 represents honest agents. The reputation value of an agent is derived by averaging all reports that concern the agent. A payment scheme for reputation reports is introduced into the reputation system to encourage agents to report truthfully about their interactions' results. According to the scheme, agents who report incorrectly have to pay an amount of money as a punishment for cheating during the process of selling reports and buying reputation information, while honest agents will not lose money. Thus, the reputation mechanism makes it rational for an agent to report its observations honestly.

Another example is the Yu and Singh system [YS02]. To manage reputation in e-commerce, Yu and Singh developed a reputation system that considers an agent's reputation as an average value of all the ratings the agent has received. The reputation system is implemented within a distributed agent architecture, and the Dempster-Shafer theory of evidence [GS90] is applied to represent and propagate ratings that agents give to each other. When examining the trustworthiness of a given partner, a peer combines its local evidence, which is based on direct prior interactions with the partner, with testimonies of others regarding the same partner. Direct experiences are considered as the primary source for determining the

reputation of the target agent, and only when direct experiences are not available, the system refers to witness information. One disadvantage found in the Yu and Singh reputation system is that it does not fully protect against spurious ratings generated by dishonest agents, because all agents are assumed to behave honestly and always give fair ratings.

3.3.3 Blurred Systems

Huynh et al. [HJS06] design a blurred system by combining reputation, context-based rules, and credentials in an agent system. This feature enables the blurred systems to provide a reputation metric with an unspecified time-dependent weight-function, where old ratings lose their influences on the current reputation over time. It is assumed that there is a high possibility that agents behave more like they did in their most recent transactions than they did in the long ago past. Based on this assumption, this metric can monitor the changes of agents' behaviors during their lifetime. The reputation of an agent $a \in A$ is

$$r(a) = \sum_{i=1}^{\#\left(\overline{E_a}\right)} \frac{\rho\left(a, \overline{E_a}[i]\right)}{\#\left(\overline{E_a}\right) - i + 1}. \tag{7}$$

The blurred systems use $\{-1, 0, 1\}$ as the set of possible rating values. Multiple ratings from the same agent are always considered. This mechanism may cause the problem that agents in the same group can manipulate their reputation values by giving each other high ratings, though low weights are assigned to old ratings.

Carter et al. [CBG02] study the blurred systems by identifying a set of roles within an information-sharing community where agents attempt to exchange relevant information with each other to satisfy other agents' requests. The reputation of an agent is based on the degree of fulfillment of roles. If agents have fulfilled their roles and get approved by the community, they are rewarded with a positive reputation, otherwise they are punished with a negative reputation. The reputation given to the agents measuring their performance in different roles is context dependent, which means the reputation defined in one community is not applicable in other communities that have different sets of roles. There is no universal way to calculate reputation for all communities. An agent's reputation is calculated as a weighted sum of the degree of satisfaction of each role. The weights are decided according to the specific community. The reputation value for each agent is calculated using a

centralized mechanism so that the global measure can be observed by all members of the community.

Schlosser et al. [SVB06] proposed a system that weights the ratings in a quadratic way to give recent ratings more power on the reputation. The purpose is to make the system stand agents who try to behave honestly to build a high reputation in the beginning and switch to a malicious behavior until their reputation gets too bad. This so-called BlurredSquared system calculates the reputation of an agent $a \in A$ as

$$r(a) = \sum_{i=1}^{\#\left(\overline{E_a}\right)} \frac{\rho\left(a, \overline{E_a}\left[i\right]\right)}{\left(\#\left(\overline{E_a}\right) - i + 1\right)^2}. \tag{8}$$

An extreme case of the blurred systems is the OnlyLast system, studied by Dellarocas [Del03]. It is assumed that an agent behaves like it did the last time, no matter what it did before. Based on the assumption, only the most recent rating given to an agent is considered, and the reputation of an agent $a \in A$ without consideration of transaction values is calculated using

$$r(a) = \rho\left(a, \overline{E_a}\left[\#\overline{E_a}\right]\right). \tag{9}$$

3.3.4 Beta System

The Beta reputation system, proposed by Jøsang and Ismail [JI02], is designed in the purpose of predicting statistically an agent's behavior in its next transaction. The system evaluates the data about an agent's previous transactions and derives the probability of whether the agent behaves good or bad. Two variables, the share of good (r) and bad (s) transactions an agent made in the past, are used as parameters for the Beta-distribution, the expectation of which shows the future behavior of the agent. They are computed as

$$r^a = \sum_{i=1}^{\#\left(\overline{E_a}\right)} \lambda^{\#\left(\overline{E_a}\right) - i \cdot \left(1 + \rho\left(a, \overline{E_a}[i]\right)\right)/2}, \tag{10}$$

and

$$s^a = \sum_{i=1}^{\#\left(\overline{E_a}\right)} \lambda^{\#\left(\overline{E_a}\right)-i\cdot\left(1-\rho\left(a,\overline{E_a}[i]\right)\right)/2}. \tag{11}$$

The reputation of an agent $a \in A$ is calculated as

$$r\left(a\right) = \frac{r^a - s^a}{r^a + s^a + 2}, \tag{12}$$

where $0 \leq \lambda \leq 1$.

According to the metric, the Beta probability density function (PDF) acts as the mathematical basis for computing ratings. The system takes positive or negative ratings as input values. The updated reputation values, also called posteriori reputation values, are computed by combining the previous reputation value with new ratings. Possible ratings are real numbers between -1(negative) and 1 (positive). By evaluating the reputation values concerning previous transactions, the reputation system derives the probability that an agent will behave good or bad in the next transaction. The share of good (r) and bad (s) transactions is calculated to show the agent's performance in the past, e.g., $r = 0.5$ and $s = 0.5$ means the agent behaves in a neutral way. These two variables are parameters for the Beta distribution that can be used to calculate the expected probability of the agent behaving in a certain way in the future.

The Beta system is a global/centralized system from the point of view of visibility types; see Section 3.1. In terms of information sources, both direct information and witness information are used for the evaluation of agents' previous performance, and are the basis of the prediction of an agent's future behavior.

3.3.5 Adaptive Systems

Sporas and Histos Sporas, proposed by Zacharia et al. [ZMM00], is a reputation system based on the following principles:

1. The reputation value of a user is never lower than the reputation of a new user.

2. Users with very high reputation values experience smaller rating changes than users with a low reputation.

3. When two users interact more than once, only the most recent rating between them is considered.

The Sporas system adopts a computational method from the Glicko system [Gli99], an approach used to evaluate the player's relative strengths in pairwise games, for measuring the reliability of users' reputation based on the standard deviation of reputation values.

The Sporas system is more robust to users' behavior changes and the reliability measures make it easier to interpret the reputation values. The major limitation of Sporas is that all new users are discriminated in the system, i.e., the reputation values of any existing users are always strictly higher than the reputation value of a beginner.

The Sporas system provides a global reputation value for each member of a community, whereas the Histos system is designed to provide a personalized reputation value [ZMM00]. The information sources in Histos include direct information and witness information. In this reputation system, direct experiences refer to the most recent experience with the agent that is being evaluated.

Zacharia et al. represent pairwise ratings using a directed graph. Similarly to the TrustNet of Schillo et al. [SFR00], in the directed graph the nodes represent agents and the edges refer to the most recent reputation rating given by an agent to another. The agent owning the graph is shown as the root node. In Histos, reputation values of agents are calculated recursively. The reputation of an agent on level L_x $(x > 0)$ of the graph is calculated as a weighted average of the ratings that agents on level L_{x-1} provided to the agent. The weights come from the reputation values of the agents who rate the target agent. Thus, an agent's reputation value is equal to the rating it has received if the agent is directly rated by the graph owner.

In Histos, the reputation value is context independent and no special mechanisms are offered to deal with dishonest agents. Another drawback of this reputation system is that it measures the reliability of a witness based on his or her reputation value. For example, one agent acting honestly as a seller or buyer does not mean it will not hide or provide biased information as a witness.

ReGreT System ReGreT [SS02] is a reputation system in which agents themselves evaluate reputation in a decentralized way. In ReGreT, each agent has the capability to evaluate the reputation of other agents. An agent rates its partner's

performance after every interaction and the ratings are recorded in a local database. The relevant ratings can be fetched from the database during the evaluation process if needed. The reputation value is derived from these ratings and calculated as the weighed mean of all ratings concerning agents' direct experiences. Each rating is weighed according to its recentness, e.g., a more recent rating is weighted more than those that are less recent.

ReGreT takes into consideration the possibility of incorrect reports from dishonest agents. Similar to Sporas, ReGreT uses a reliability value that represent the predictive power of each reputation value. The reliability value is calculated using two measures: the number of ratings used in calculating a reputation value and the deviation of these ratings.

ReGreT incorporates all of the information sources, i.e., direct experiences, witness information and sociological information, discussed in Section 3.1. Based on the assumption that agents are willing to share their opinions about one another, ReGreT provides a witness reputation component involving a method for aggregating witness reports, which also takes into account the possibility of dishonest reports. The component operates on the social network built up by each agent to find witnesses, to choose which witnesses to be inquired, and how to weight the witnesses' reports. However, the way to form such social networks is not explained in [SS02]. ReGreT also refers to sociological information, such as neighborhood reputation and system reputation, as source of information to decide an agent's overall reputation. The neighborhood reputation is calculated from the reputation of the target agent's neighbor agents based on fuzzy rules, which also requires a social network to work. The system reputation is a default reputation value assigned to the target agent based on its social role in an interaction, e.g., a buyer or a seller.

Perseus System Perseus [Nur07] is a personalized reputation system that aims to overcome the problems existing in most centralized systems for online marketplaces. One example of the problems is that agents can behave honestly in the very beginning, but they may later abuse their high reputation and act dishonestly if they know their reputation. In Perseus, an agent's reputation depends on the agent who queries it. The reputation value is computed taking into account two information sources: first hand reputation, i.e., personal direct experiences, and third party reputation, i.e., witness information. The direct experiences act as the main source of information, whereas witness information are used when agents have no interac-

tion history. These two reputation values are aggregated into a single value using a weighted linear sum, and the weights of the linear sum, called coefficients, are updated over time using a stochastic approximation algorithm. The approximation algorithm increases the coefficient for a particular source towards one whenever the reputation source and the rating agree, and respectively, the coefficient is decreased towards zero when a disagreement about the rating and the reputation source happens. To this end, the Perseus system, together with the Sporas system and the ReGreT system described above, can be regarded as an example of adaptive reputation systems.

In the Perseus system, the possible ratings an agent may receive from others are $\{-1, 0, 1\}$. The reputation value of an agent a is a combination of two estimates: the first hand reputation (personal experiences) and third party reputation (witness reports) [Nur07]. The two variables can be denoted as $\gamma_{a,b}^t$ for the first hand reputation of agent a from the point of view of agent b after t transactions, whereas $\phi_{a,b}^t$ for agent b's view of the third party reputation of agent a. The first hand reputation $\gamma_{a,b}^t$ is computed using a stochastic approximation algorithm [STA05]:

$$\gamma_{a,b}^t = \gamma_{a,b}^{t-1} + \alpha \left(\rho \left(a, e_{a,b}^t \right) - \gamma_{a,b}^{t-1} \right), \tag{13}$$

where $\alpha = max \left[\frac{1}{t}, 0.001 \right]$.

The third party reputation of agent a is calculated using the average sum of ratings from agents that are considered trustworthy, i.e.,

$$\phi_{a,b}^t = \frac{1}{\#T} \sum_{i=1}^{t} \left(\gamma_{a,b}^i \right). \tag{14}$$

Perseus uses a weighted linear sum to aggregate first hand and third party reputation to compute the trustworthiness of agent a from the point of view of agent b, which is

$$r\left(a\right) = \frac{\pi_1}{\pi_1 - \pi_2} \gamma_{a,b}^t + \frac{\pi_2}{\pi_1 - \pi_2} \phi_{a,b}^t. \tag{15}$$

Here, π_1 and π_2 are weights for first hand and third party reputation respectively. According to rules described in Section 3.3.5, each coefficient $\pi_i \, (i = 1, 2)$ is updated using

$$\pi_i^{t+1} = \pi_i^t + \begin{cases} \beta\,(1.0 - \pi_i^t) & \text{if agree} \\ \beta\,(0.0 - \pi_i^t) & \text{if disagree} \end{cases}, \tag{16}$$

where β is the step size.

3.4 Summary

The reputation systems described in the previous section are summarized in Table 2 from the point of view of the classification dimensions discussed in Section 3.1. The rating scale is also considered as one of the characteristics of reputation systems. These systems are grouped according to different types of metrics they use.

It is important to note that though two classification aspects are described to allow comparing different reputation systems, these classification dimensions do not always capture differences with the reputation systems. In some cases, the classification for a specific reputation system in one or another category is subject to our own understanding. There are also other classifications that can be used to categorize reputation systems (see, e.g., [SS05]), but these two dimensions are closely related to the scope of our study.

Table 2: A comparison table showing the characteristics of reputation systems. The table uses the following abbreviations: DE for *direct experiences*, WI for *witness information* and SI for *sociological information*.

System	Example(s)	Information Source	Visibility Type	Rating Scale
Accumulative	eBay system	WI	Global	$\{-1, 0, 1\}$
Average	Jurca and Faltings [JF03]	WI	Global	$\{0, 1\}$
	Yu and Singh [YS02]	DE, WI	Personalized	$\{-1, 0, 1\}$
Blurred	Blurred system [HJS06]	WI	Global	$\{-1, 0, 1\}$
	BlurredSquared system [SVB06]	WI	Global	$\{-1, 0, 1\}$
	Carter et al. [CBG02]	WI	Global	$\{-1, 0, 1\}$
	OnlyLast system [SVB06]	WI	Global	$\{-1, 0, 1\}$
Beta	Beta system [JI02]	WI	Global	$[-1, 1]$
Adaptive	Sporas [ZMM00]	WI	Global	$\{0.1, ..., 1\}$
	ReGreT [SS02]	DE + WI + SI	Personalized	$\{-1, 0, 1\}$
	Perseus [Nur07]	DE + WI	Personalized	$\{-1, 0, 1\}$

4 Multi-Agent Simulation of Reputation Systems

As discussed in Section 2.4, real world systems can be modeled and observed using multi-agent simulation. In order to learn how to form a specific simulation environment for evaluation of reputation systems, this section reviews the multi-agent approach (Section 4.1); the ART testbed, especially the multi-agent simulation framework (Section 4.2); and simulation scenarios developed for studies of reputation systems in different domains (Section 4.3).

4.1 Multi-Agent Approach

The multi-agent approach is considered a combination of several disciplines, of which the two most important ones are *distributed artificial intelligence* (DAI) and *artificial life* (AL). The work of DAI started in the United States in the early 1980s [Syc98] aiming to create organizations of systems capable of giving distributed solutions for complex problems requiring intelligence.

In early years, open systems have been extended beyond DAI, by regarding problem solving as the activity of several experts and viewing a reasoning process as a sequence of choices [Hew86, Hew91]. These ideas are regarded as bases for the creation of multi-agent platforms. In contrast to the work on DAI, which considers that intelligence proceeds from the manipulation of symbols, artificial life emphasizes on viability, behavior and autonomy. AL approaches focus on understanding cooperation and coordination mechanisms on the basis of criteria not involving the intervention of symbols. These areas are equally important and have influenced the development of multi-agent systems (MASs) on aspects of communications, automation and robotics on the regulation of actions in a real world.

Multi-agent systems have several areas of applications. Generally speaking, MASs are able to:

1. solve problems that are too complicated for a centralized agent to deal with due to resource limitations;

2. allow for inter-operation and interconnection between multiple existing systems;

3. solve problems that naturally concern a society of autonomous interacting agents;

4. provide solutions using spatially distributed information sources; and
5. enhance performance, e.g., computational efficiency, reliability, extensibility, maintainability, responsiveness, and flexibility.

Four main characteristics of MASs are:

1. no system global control;
2. decentralized data;
3. asynchronous computation; and
4. each agent has a limited viewpoint, e.g., incomplete information or limited capabilities for solving problems.

Research in MASs is related to the study, behavior and construction of an assembly of possibly pre-existing autonomous agents that interact with each other. An MAS is a loosely combined network of problem solvers, i.e., agents who are autonomous and heterogeneous in nature, that interact to solve problems that are beyond the individual capabilities or knowledge of each agent [Syc98]. According to Wooldridge and Jennings [WJ95], an agent usually has the following properties:

1. Autonomy. Agents have direct control over their actions and internal states so that they can operate without the direct intervention of humans or others.
2. Social ability. Agents can interact with other agents (and possibly humans) using certain agent-communication language.
3. Reactivity. Agents perceive their environment, e.g., a physical world consists of a collection of other agents or a simulated world with other agents, and respond in a timely fashion to changes that occur in it.
4. Pro-activeness. Agents are able to react to their environment and exhibit goal-directed behavior by taking the initiative.

4.2 ART Testbed

ART stands for "Agent Reputation and Trust" that indicates the problem domain of the testbed. It is designed not only for testing a single reputation system in different communities, but for providing a framework to allow different agents using different trust and reputation mechanisms to compete with each other. It is necessary for us to take an insight into how the ART testbed works because the design purposes of our simulation are quite similar to that of the ART testbed. However, the ART testbed cannot be directly used in our work, since it was developed so extensively to meet the demands from competitions between trust and reputation mechanisms. Instead, we focus on the research objectives of the testbed that can be considered as requirements for our simulator design. In addition, our simulation engine described in Section 5.3 is implemented based on the ideas from the testbed's architecture.

Section 4.2.1 covers the basics of the ART testbed. Section 4.2.2 represents the research objectives and design requirements proposed by the ART team who developed the ART testbed. Section 4.2.3 describes the architecture of the ART testbed especially the work processes of its simulation engine.

4.2.1 Basics

Trust and reputation in multi-agent systems is a hot topic of research in both industry and academia, and a wide variety of trust and reputation modeling technologies and many metrics for empirical validation have been developed. However, it is difficult to find a suite of problem instances to represent the topic as a whole. Unified performance metrics and objectives for trust technologies must be provided to the public to evaluate systems based on transparent and recognizable standards [FKM+05a]. Barber et al. [BFK03] argue that objective standards are essential to justify successful trust and reputation systems and to give a starting point of research strategies for future work. As a solution, a working group named ART was created to develop a testbed for experimentation and comparison of different mechanisms to foster a wide range of trust and reputation research problems using unified experimentation methods [FKM+06].

The Agent Reputation and Trust (ART) testbed[14] was presented for the first time during AAMAS'05[15], and since AAMAS'06, the competition of trust and reputation

[14] `http://www.art-testbed.net/` [Retrieved 2009-09-09]

[15] AAMAS is the International Conference on Autonomous Agents and Multi-agent Systems.

mechanisms using the ART testbed has been held annually.

4.2.2 Requirements

According to Fullam et al. [FKM+05a, FKM+06], the ART testbed provides solutions to fulfill trust and reputation research objectives, such as:

1. Agents have to interact with others to gain resources like physical goods, information or services.
2. Agents should have the capability to decide whether interactions are risky, e.g., the agreements between agents may or may not be fulfilled.
3. Agents should be able to minimize risks, for instance, to interact with reliable agents by predicting if they are most likely to fulfill agreements.
4. Agents should be able to make these predictions via trust and reputation models.

Concerning the decision making abilities as part of the requirements for simulation setup, an agent should be able to:

1. **Identify and isolate untrustworthy agents.** An agent must be able to identify malicious agents and refuse interactions with them in order to protect itself from exploitation.
2. **Evaluate the utility of an interaction.** An agent must be able to evaluate the utility of an interaction before deciding to participating in the interaction. This is necessary because the agent can get better negotiate payment after knowing the possible utility.
3. **Determine whether to interact with other agents.** An agent must be able to decide whether to interact with other transaction partners or not. In order to make the decision, the agent needs to predict whether the partner will fulfill the agreement. The prediction can be made by comparing the number of successful transactions (decisions) made by the agent with the total number of transactions initiated by the agent.

For the purpose of our simulation design, two requirements are added with specific focus on agents' capability to effectively interpret their behavior model (Section 6.1) into decision making of giving feedback (rating) to their interaction partners. In other words, each agent must be able to:

1. **Decide whether to give feedback to its partner** After an interaction is finished, an agent must be able to choose whether to give feedback to its interaction partner. This decision is made depending on the agent's own behavior model. For example, an honest agent will always give feedback to its interaction partner, whereas a selfish agent will never do that.

2. **Decide what kind of feedback to give to its partner** Given willingness to share opinion on an completed transaction with other agents, an agent must be able to decide what kind of feedback to give to its interaction partner. The feedback can be either correctly reflecting or completely different from the utility that an agent gains from an interaction.

On the other hand, our simulation aims to evaluate a set of reputation systems in a virtual environment, i.e., a widget sharing community. From the methodological perspective, evaluation criteria are needed for examining results of simulations and determining the strengths and weaknesses of different reputation systems. The research requirements for reputation systems presented in [FKM+05a] are taken into consideration for the evaluation criteria selection (Section 6.1.3). In terms of modeling reputation systems, a system should be:

1. **Accurate.** A reputation system should be able to make accurate predictions about another agent's future behavior. An indicator for accuracy can be the similarity between the agent's predicted behavior and its true behavior.

2. **Adaptive.** An agent's behavior should be dynamic. It may suddenly lose competence or behave maliciously. A reputation system should be adaptive to reflect these behavior changes.

3. **Quickly Converging.** A reputation system should quickly adapt to the changes of agents' behavior.

4. **Multidimensional.** A reputation system should reflect an agent's trustworthiness characteristics across multiple categories, e.g., an agent can act as both a buyer and a seller in online marketplaces.

5. **Efficient.** Algorithms used in reputation systems should be efficient in terms of computational time.

Not all of the requirements for reputation systems are relevant to our simulations. In particular, *multidimensionality* is not one of our research goals, because in our case, each agent acts as either a user or a developer. We return to this topic and describe the evaluation criteria in our simulation in Section 6.1.3.

4.2.3 Architecture

The testbed can be used for both competition and experimentation. For the purpose of competition, the ART testbed is designed as less domain-specific to avoid restricted solutions. In addition, the testbed provides a suite of tools with flexible parameters, allowing researchers to execute customizable and repeatable experiments with it. Therefore, ART can be used to compare different reputation systems using objective metrics [FKM+05a].

To better understand how the ART testbed works, we take an insight into its implementation architecture. Fullam et al. [FKM+05b, FKM+05c] list five components that are included in the testbed architecture:

1. Game Server, which is designed for setting up and running games;
2. Simulation Engine, which is responsible for controlling the simulation environment by enforcing chosen parameters;
3. Agent Skeleton, which assists game players to develop their code;
4. Database, in which Simulation Engine calculations and experiments analysis results are stored; and
5. User Interfaces (Game Setup Interface and Game Monitor Interface), which help players to set up and view their games.

The interactions between the system components are illustrated in Figure 5. This figure shows that the Simulation Engine serves as the heart of the testbed. It is responsible for initiating a simulation game, controlling the simulation environment using various parameters, allocating interaction pairs and coordinating communication among interaction pairs. In each period, the Simulation Engine manages

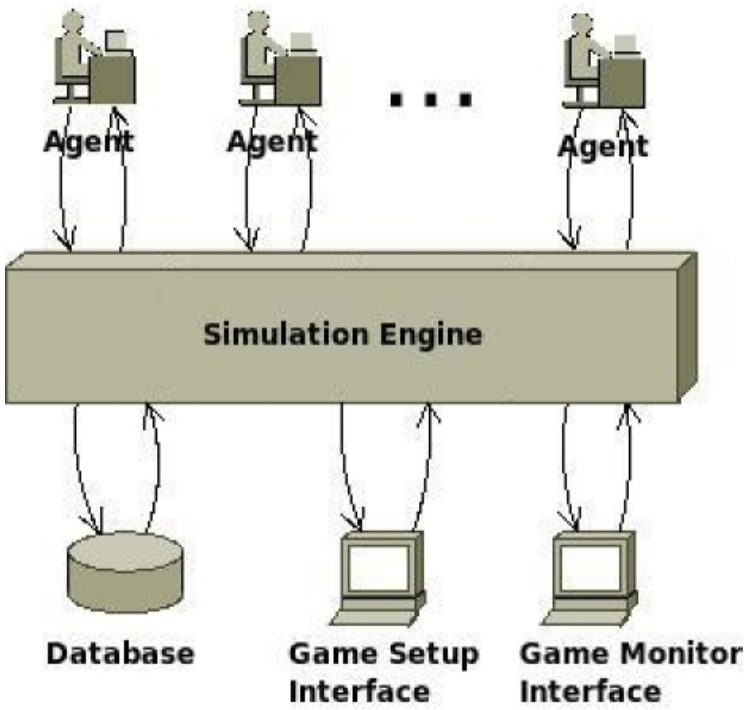

Figure 5: The ART testbed architecture. Modified from [FKM+05c].

agents' actions and decision making, transactions, calculation of final reputation values, and distribution of reputation values. Through the Simulation Engine, the Database collects and stores simulation environment and agent data about ratings and calculations of overall reputation values. The User Interfaces graphically display details about the simulation progress to make them viewable. The Agent Skeleton is in charge of coordination tasks with the Simulation Engine, such as formalizing opinions and calculating reputation values. In addition, the Agent Skeleton allows customization of algorithms and implementation of new algorithms to handle communication and interactions among agents.

To illustrate how agents' actions are coordinated in the Simulation Engine, a custom scenario is formed in the art appraisal domain. In the art appraisal community, agents can act as either painting appraisers with different levels of expertise in different artistic areas, e.g., classical, postmodern, or as clients who ask for appraisals for paintings from various aspects. An appraiser can give the appraisal based on her own expertise, or purchase opinions from other appraisers, if the appraiser is not sufficiently knowledgeable. Details about the art appraisal domain rules can be found in [FKM+05a].

Given the scenario, the Simulation Engine coordinates agents' actions based on the

following workflow:

1. In the beginning, the Simulation Engine assigns clients to each appraiser. To simulate real world clients' preference for correct appraisers, the more accurate appraisals an appraiser gave in the past, the larger shares of clients will be allocated to the appraiser by the Simulation Engine. To simulate clients' requests for appraisals, the Simulation Engine sends notifications of appraisal requests to the allocated appraisers. It also pays fees into correspondent appraiser's bank account to simulate the pre-payment of appraisals from clients.

2. The Simulation Engine then enables reputation transactions among appraisers. Appraisers may check reputation information about third-party appraisers' expertise in given areas to decide from which appraisers to buy appraisal opinions. When an appraiser requests reputation information, the potential provider may reject the request, which directly aborts the transaction, or accepts the request and sends the reputation to the requesting appraiser after receiving payment. However, the reputation information from the provider need not be correct. The transaction is aborted if the provider chooses to reject the request. The Simulation Engine is responsible for handling the passing of transaction messages. Moreover, the Simulation Engine handles payments between appraisers by transferring money from the requesting appraiser's bank accounts into the provider's bank accounts.

3. The Simulation Engine also supports opinion transactions between appraisers. If an appraiser sends request to a potential opinion provider, the provider can reject the request, thus abort the transaction. Alternatively, the provider may choose to accept the request. Upon receipt of the potential opinion, the requester can either decline it to abort the transaction or pay the fees. After receiving the payment, the provider sends the opinion, which may be untruthful. Similar to the previous step, the Simulation Engine controls the passing of transaction messages and the payment transfer.

4. Finally, the Simulation Engine calculates the appraiser's final appraisal using a weighted average of the opinions the appraiser has bought. Opinion weights are real values between zero and one that an appraiser assigns to another agent's opinion.

Table 3: Payoff matrix in "win-lose" terminology for the Prisoners' Dilemma game, where C = *cooperate* and D = *defect*.

	C	**D**
C	win - win	lose much - win much
D	win much - lose much	lose - lose

4.3 Simulation Scenarios

As reputation systems have been studied in different domains, in order to capture all the relevant aspects, many of the attempts have been made to devise a generic enough scenario but few of them have been successful [FKM+05a]. As a result, the Prisoners' Dilemma (Section 4.3.1) and customized scenarios (Section 4.3.2) have become two popular ways that are used for simulation of reputation systems to solve this problem.

4.3.1 Prisoners' Dilemma

The *Prisoners' Dilemma* is a game in which two players as a pair can each choose to cooperate or defect in order to gain better utilities. Fudenberg and Tirole [FT91] describe the Prisoner's Dilemma game as follows:

> Two suspects are arrested for a crime. The police have to convince the suspects to give testimony against each other, because there lacks sufficient evidence to convict either suspect. The suspects are separated in different cells to prevent them from communicating with each other. The police tell each suspect that if he testifies against (does not cooperate with) the other, he will be released and will receive a reward, only if the other does not testify against him. If neither of the suspects testifies, both of them will be released due to insufficient evidence. If one testifies, the other will go to prison. If both of the suspects testify, they will get the rewards for testifying.

In the Prisoners' Dilemma game, two players act as a pair, who have two possible actions: to *cooperate* or to *defect*, i.e., to testify. Table 3 shows the payoff matrix for the Prisoners' Dilemma game.

The equilibrium analysis of the Prisoners' Dilemma can be performed using *elimination of dominated strategies* [FT91]. The analysis results show that the unique Nash equilibrium of the Prisoners' Dilemma, i.e., the strategy *(defect, defect)*, can bring short-term maximized utilities to both players, whereas the long-term optimal strategy prefers *(cooperate, cooperate)*.

The Prisoners' Dilemma can be seen as a generic scenario that allows simulation studies in which conditions cooperation exists. [MMH02] is one of the examples that apply the Prisoners' Dilemma as a simulation scenario to experimental evaluation of reputation systems. However, multi-agent simulation using the Prisoners' Dilemma scenario have several problems, e.g., agents have no opportunity to isolate untrustworthy opponents, because they have to interact with all agents in the community [FKM+05a].

4.3.2 Custom Scenarios

Custom scenario is another alternative that can be used in simulation-based evaluation of reputation systems. Examples of custom scenarios for evaluation purposes of reputation systems in different domains are:

1. A scenario of a supply chain with many markets that allows complex settings for agents to make their own decisions on various transactions of buying and selling [SS02].

2. A scenario of a community where agents provide homogeneous symmetric services to each other [SVB06].

3. A scenario of art appraisal community for simulation on ART testbed; see Section 4.2.3.

Other examples of custom scenarios can be found in domains, such as P2P application, e-commerce and multi-agent systems [GJA03, XL03, Nur07]. However, there still lacks a suitable scenario designed for simulation of widget sharing reputation systems. As the issue has close connections to our simulation setup, we return to this topic and describe our simulation scenario in Section 5.

5 Simulation Setup

The reputation systems discussed in Section 3 have been simulated and evaluated in different contexts. However, none of the previous studies has created a scenario in which agents (i.e., users and developers) interact with each other indirectly via a transaction object. In this simulation, we consider an online widget sharing community where developers provide widgets and users act as consumers of virtual products. This section describes a custom scenario in Section 5.1, followed by a regular user model and a developer model in Section 5.2. The implementation of the main workflow of our simulation is presented in Section 5.3, and Section 5.4 covers the simulation parameters that are used in the simulation setup.

5.1 A Custom Scenario

The custom scenario is built around the idea of emulating a real world widget sharing system. In widget sharing systems, a set of agents interacts with each other indirectly, i.e., via a piece of software. In this system, some agents act as *users* that download widgets, which were developed by other agents who are known as *developers*. The design of a custom scenario is based on this structure and oriented to the study of reputation systems.

Consider the following scenario:

> A developer Joe publishes a Sudoku widget in a widget sharing community, called WidSets.com. Later user Jane browses the WidSets.com webpages and finds the Sudoku widget. She checks this widget's rating and information concerning the developer, Joe. After evaluating the information, Jane decides to download the widget. After using Sudoku for a while, she decides to share her own opinion with other users by rating it. A reputation system working behind the community aggregates the rating from Jane into Sudoku's overall rating and makes it visible to other users of WidSets.com. The reputation system also distributes this rating to Sudoku's developer, Joe, and aggregates it into Joe's overall rating.

In this scenario, there are two kinds of agents: regular users and developers. It is important to note that, in reality, not all users and developers are necessarily honest,

and this needs to be taken into consideration in the simulation. Developers can provide malicious widgets, whereas some users may give dishonest ratings to widgets. To clarify it, several decision making phases are modeled within an interaction for both users and developers.

5.2 Decision Making

The decision making phases are described for the regular users (Section 5.2.1) and developers (Section 5.2.2) respectively.

5.2.1 Regular User

Figure 6 shows an overview of the interaction process from the point of view of a regular user. This process contains a sequence of three decision making phases. These phases are:

1. Should I download the widget? As it suggests, this question is asked after a certain widget has been selected by a user; see Section 5.3. The user has to decide whether to continue a download transaction or not. This decision is based on a behavior model which takes into account the rating of the widget and its developer. She may accept and download the widget or abort the transaction.

2. Should I rate the widget? If the user decides to download the widget, the nature of the widget (good or malicious) is revealed to the user who can share her opinion of the widget with other users in the community. The user can also choose to skip the rating step.

3. What kind of rating should I give to the widget? This is a further question when the user chooses to rate the widget she used. The rating decision depends on the outcome (i.e., the nature of the widget), and the user's behavior models; see Section 6.1.1.

5.2.2 Developer

Figure 7 shows the interaction process from the developer's point of view. Similarly, developers also have to make decisions at some point of the whole interaction pro-

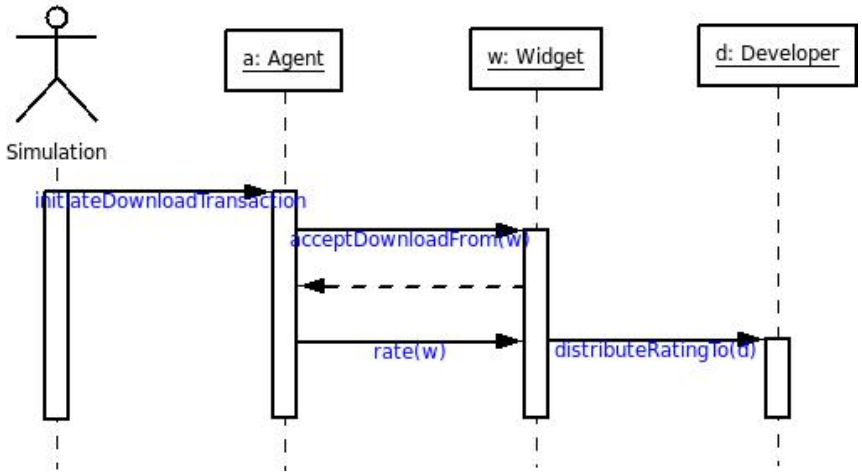

Figure 6: An overview of the interaction process from a user's point of view. Within each interaction with a selected widget, the user has to make decisions in three phases: (i) whether to download the widget or not; (ii) whether to rate the widget or not; and (iii) to give what kind of rating to the widget.

cess. However, since they cannot control the rating process itself, the main tasks of developers are to decide:

1. Should I upload a widget to the online community? Here developers have to decide whether to create a new widget depending on their willingness to contribute more widgets to the community. For each developer, the number of widgets he has may influence this choice.

2. What kind of widget should I create? Developers make this decision according to their behavior models; see Section 6.1.1.

5.3 Implementation

To handle the simulation environment with a custom scenario proposed in Section 5.1, we implemented a simulation engine that is adapted from ART testbed's architecture described in Section 4.2.3. The simulation engine is responsible for initiating a certain number of developers, users and widgets, and coordinates interactions among them. Figure 8 shows the simulation engine's workflow from the point of view of implementation.

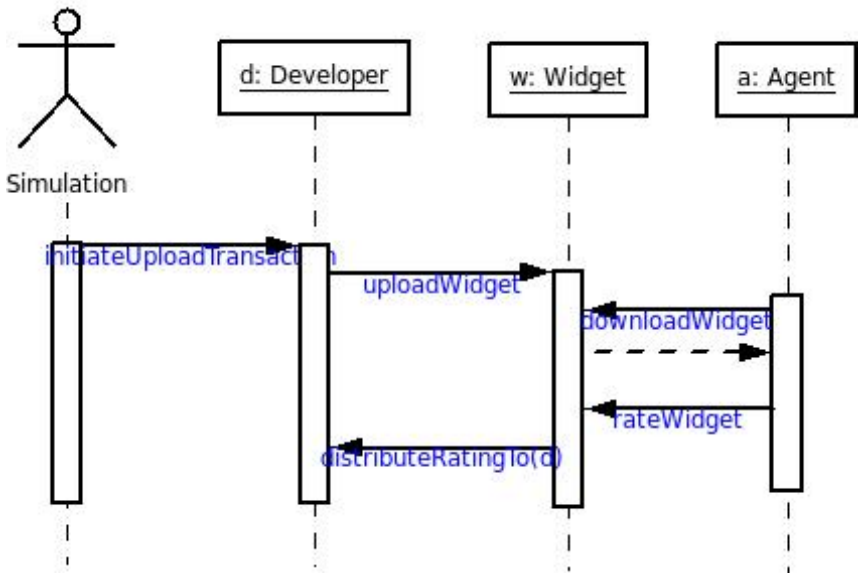

Figure 7: An overview of the interaction process from a developer's point of view. In the *uploadWidget* step, the developer has to make decisions in two phases: (i) whether to upload the widget or not; and (ii) to upload what kind of widget.

Initiation The simulation engine begins its work by initiating a certain number of developers, users and widgets. To simulate developers and users in the online widget sharing community, the engine generates a user pool and a developer pool. Each agent (i.e., user or developer) is assigned a behavior model, which influences the agent's decisions throughout the simulation. To simulate widgets owned by developers, the engine produces widgets for each developer. These widgets are assigned a different nature (i.e., good, neutral or malicious) and stored in a widget pool.

To make the simulation realistic, the simulation engine decides the number of users (or developers) for each behavior type, the number of widgets each developer owns, and the nature of widgets using probabilistic distributions learned from the empirical data; see Section 5.4.

Iterations After initiation, our simulation proceeds as a series of iterations. The engine manages adding new widgets into the widget pool, selecting widgets for users, collecting ratings from users, and calculating overall ratings for widgets and developers. Within one iteration, these tasks are divided into four phases and the simulation engine controls these phases in sequence:

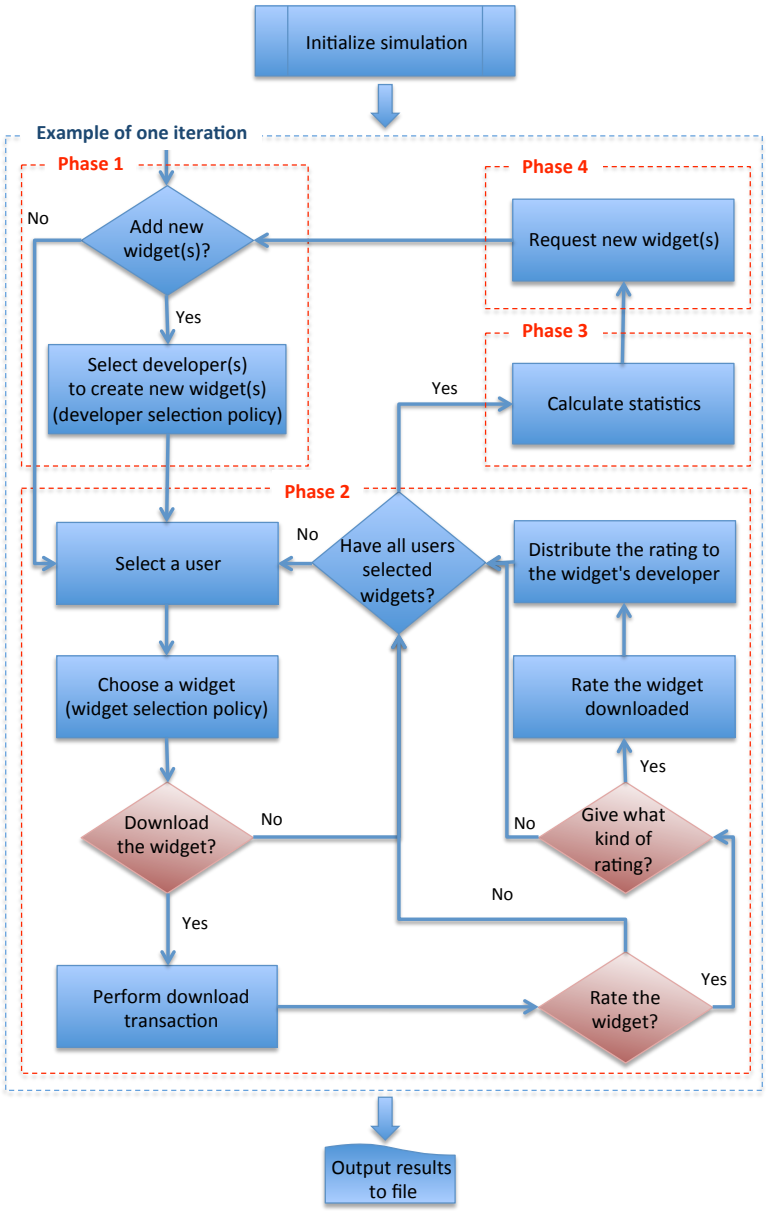

Figure 8: A flowchart shows how our simulation engine works during one simulation iteration. The workflow includes four phases: adding new widgets (Phase 1), transacting and rating process (Phase 2), calculating final reputations of widgets and developers (Phase 3) and requesting for new widgets (Phase 4). The red diamonds concern users' decision making process.

Phase 1: At the beginning of each iteration, the simulation engine checks if there is a request[16] for new widgets. If there is a request, the engine selects developers from the developer pool to create new widgets. The developer selection policy obey three rules: (i) one developer can only produce one new widget in one iteration, (ii) the candidate has not generate any widgets during the previous iteration, and (iii) the number of widgets the candidate owns must always obey the probabilistic distribution defined specifically for this simulation parameter; see Section 5.4.3. After developer candidates are selected, they are responsible to create widgets and these widgets are added into the widget pool. If no new widgets are required, the engine will enter the next phase.

Phase 2: In this phase, a loop starts with selecting a user candidate from the user pool. The simulation engine chooses a widget for the user based on the widget selection policy, i.e., the widgets with higher popularity are more likely to be chosen. The user, then, have to make several decisions that are shown as red diamonds in Figure 8. As users' decision making process has been discussed in Section 5.2.1, only the "yes" paths are gone through here. As the user decides to download the widget, the simulation engine performs the download transaction and sends a signal to trigger the rating process. Based on the decisions made by the user, the engine gives a reputation score to the widget and distribute the rating to the widget's developer. The loop ends only when there is no user left unselected in the user pool.

Phase 3: For the purpose of experiment analysis, all the ratings are collected and calculated for widgets and developers respectively.

Phase 4: To make our simulation close to the real world situation, the simulation engine is responsible to request for new widgets by the end of one iteration. The engine generates the number of new widgets following a pattern of events that occurs with an average rate and time independently, i.e., the Poisson distribution. The setting of variables is described in Section 6.1.

[16]This request is generated in the fourth phase during the previous iteration, so the first phase is usually skipped during the first iteration.

5.4 Simulation Parameters

The simulation implementation described in the previous section depends on various parameters. Table 4 gives an overview of the parameters that are specifically concerned with the simulation setup. To ensure that the results of our simulation are realistic, we initialize these parameters using statistical models that are determined by analyzing empirical data collected from an online widget sharing website.

Section 5.4.1 explains the methodology that is used to find the best models. Section 5.4.2 explains how the data was collected from a real widget sharing website, and Section 5.4.3 illustrates the best model selection process for each parameter based on the collected data.

Table 4: Simulation Parameters

Parameter	Description
Number of users (Number of developers)	This parameter fixes the size of users (developers) for each behavior type separately at the beginning of every simulation run.
Widgets per developer	This parameter determines the number of widgets each developer has in the initialization phase.
Widget popularity	This parameter sets download frequency for each widget.

5.4.1 Methodology

Model selection is a mechanism that allows selecting the best model from a set of predefined candidates by estimating the probability of the models from empirical data. For model selection, the previous work usually focuses on statistical models of which the most popular target distribution model is the power-law [EPW+07], because many man-made and natural phenomena, such as city population, traffic in websites and earthquake magnitudes, are distributed according to a power-law distribution [GMY04, New05]. Power-law distributions are normally detected using log transformation (LT) [BWW85], but [VP06] and [EPW+07] argue that log transformation is not reliable for choosing between distribution models, i.e., the power-law and the exponential. In addition, [EPW+07] suggests that Akaike weights can be

used as measures for selecting the best statistical model that fits a given data set.

To determine the appropriate statistical models for the simulation parameters, we adopt model selection methods, more specifically Akaike weights. However, to avoid overfitting the data with unnecessarily complex models, we extend the test subjects by simultaneously considering other distributions of different complexity as candidates for our model selection. The following explains what are Akaike weights and gives an overview of the candidate statistical models tested using Akaike weights.

Akaike Weights *Akaike weights* is a model selection method that allows ranking and weighting different models by providing each model a relative weight of evidence [BA02, EPW+07]. The Akaike weights can be interpreted as the probability that a given model is the best model, given the data and the set of statistical models. To determine the best distributions to initialize simulation parameters, the Akaike weights method is adopted to select the optimum statistical model from several candidates.

Akaike weights are calculated using the difference in Akaike's information criterion (AIC) for each candidate compared with the minimum AIC [BA02]. The weights are relative likelihoods of each model, calculated by

$$w_i = \frac{\exp(-\Delta_i/2)}{\sum_{i=1}^{n} \exp(-\Delta_i/2)}, \tag{17}$$

where Δ_i is AIC difference given by $\Delta_i = AIC_i - AIC_{min}$.

The Akaike's information criterion for each model in a set of candidates is given by

$$AIC_i = -2\log[L_i(\hat{\theta}_i|D)] + 2K_i. \tag{18}$$

Here $\log[L_i(\hat{\theta}_i|D)]$ is the log-likelihood of a particular parameter value in model i given the data set D and parameters $\hat{\theta}_i$. The variable K_i considers the number of estimated parameters included in model i. The log-likelihood of the model given the data reflects the overall fit of the model (smaller values indicate worse fit). The best model is the one with the highest value of Akaike weight.

Candidate Models As discussed above, appropriate pre-defined candidate models act as the basis of model selection. Since many physical and social patterns follow a power-law distribution (Section 5.4.1), a power-law distribution model is

considered as one of the candidate models. The probability density function for a power-law distribution is [New05]:

$$p(x) = Cx^{-\alpha}. \tag{19}$$

Here C is a normalization constant:

$$C = (\alpha - 1)x_{min}^{\alpha-1}, \tag{20}$$

and the exponent α is calculated using

$$\alpha = 1 + n\left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{min}}\right]^{-1}. \tag{21}$$

The variables x_i, $i = 1...n$ correspond to the measured values of the given data set x, which can be the number of *widgets per developer* or *widget download frequencies*. Finally, the variable x_{min} is the minimum value of x. The exponent α in Equation 19 should be a positive number ($\alpha > 0$), whereas in Equation 20, the constant C is valid only when $\alpha > 1$; see [New05].

Many quantities with highly right-skewed distributions do not necessarily follow a power-law distribution [New05]. For this reason, two related distributions, the exponential and the log-normal distribution, are included in the analysis; see, e.g., [Mit04]. To ensure that the data is not overfitted with unnecessarily complex distributions, we also included a set of common distributions in the analysis; see Table 5.

5.4.2 Description of Data

To form our simulation with real world data, the Widget Library of WidSets.com was crawled. The crawling resulted in webpages for 7203 widgets and 2942 developers using a recursive crawling tool. The crawling process was initiated from a list of widgets that are ranked according to their popularity (i.e., download frequency), and from there on, the crawler recursively downloads the description pages of every widget on the list. For each widget crawled, the key information that were needed for further analysis (Section 5.4.3) was extracted, e.g., the rating and the popularity of a widget. Specifically for developers, the URLs are kept in a list and explored recursively to get essential data, such as a list of widgets each developer has uploaded.

Table 5: Summary of Akaike weights calculated for each simulation parameter. The candidate models are listed as row headers, whereas the simulation parameters in column headers.

#	Model	Simulation Parameter	
		Widgets per developer	Widget popularity
1	Exponential	0	0
2	Log-normal	$7.3 \cdot 10^{-30}$	≈ 1
3	Negative binomial	0	0
4	Normal	0	0
5	Pareto	≈ 1	$2.2 \cdot 10^{-288}$
6	Poisson	0	0
7	Two component Gaussian mixture	0	0

For most of the widgets, I only managed to crawl a small fraction of the actual users from the widgets' webpages, because (i) for each widget, only five of the most recent users are shown on the webpage; (ii) some users prefer to protect their personal information from being exposed to the public; and (iii) no user list is available from the WidSets.com website.

5.4.3 Analysis

The analysis of the empirical data described in Section 5.4.2 aims to determine the best models for each simulation parameter. Before analyzing the given data set, we removed widgets for which the number of downloads is not feasible or realistic for simulation purposes. The removed widgets are mainly pre-selected ones that users obtain when they register for the website. The following sections describe the analysis for the simulation parameters listed in Table 4.

Users and Developers Since not all the users' information is available from WidSets.com (Section 5.4.2), it is assumed that the number of users equals the

download frequency of the most popular widget, i.e., Wikipedia. Thus, millions of users ($\approx$ 5 million users) are consided as having used WidSets.com service. However, this number of users is unnecessarily large and somewhat infeasible for simulation purposes. On one hand, a large fraction of users does not download any widgets beyond the pre-selected widgets in the registration phase. On the other hand, storing information about millions of users requires significant amount of memory and this workload can largely slow down the simulation performance. To ensure that the number of users is both realistic and feasible, the users who passively accept the pre-selected widgets and do not make their decisions independently are removed.

In our simulation, a developer is defined as a person who has contributed at least one widget to the system. Since no evidence shows developers are influenced by other factors when they decide to make contributes to the online widget sharing community, we regard all of the developers to be independent decision makers and directly use the number of developers from the data; see Section 5.4.2. Moreover, we remove the pre-selected widgets from the system and achieve an appropriate ratio of users to developers.

Given the corrected ratio of users and developers, the probability distribution for each type of agents (i.e., users and developers) should be decided in the simulation. In our case, the probability specifies the likelihood that each of user and developer acts as a pair in a transaction. A Normal distribution [Rob95] is used to model the numbers of users and developers of each behavior model, with the mean as an average number of a certain type of users or developers, and a standard deviation to achieve some variation.

Widget Upload Frequencies for each developer To determine a suitable upload policy for the simulation, the upload frequencies for each developer should be analyzed. We started by plotting the number of widgets each developer owns in the widget sharing community; see Figure 9. The figure shows the number of widgets a developer has uploaded on the x-axis, and the number of developers with the same number of widgets on the y-axis. The shape of the plot shows that the number of uploads follows a heavy-tailed distribution. For this reason, a power-law or exponential distribution could be expected to fit the given data set. The analysis supports our hypothesis as the Pareto distribution, i.e., power-law distribution, has the highest Akaike weight, which is close to 1 ($w \approx 1$).

To verify the result, a log-log histogram [EPW$^+$07, New05] is used to reveal the

power-law form of the distribution. The straight line in Figure 10 suggests that the data set, i.e., the number of widgets per developer has, follows a power-law distribution. With these evidences, we select the Pareto distribution for this parameter.

Widgets Download Frequencies To determine an appropriate download policy for the simulation, we plotted the download frequencies of widgets; see Figure 11. The download frequency for each widget is shown on the x-axis, whereas the number of widgets which have been downloaded for certain times is shown on the y-axis. Since most of the points are located between 0 and 3500, the plot is truncated along the x-axis from 3500 onwards to make the shape of the distribution more clear. The shape of the plot suggests that the number of downloads follows a heavy-tailed distribution. The analysis indicates that the log-normal distribution, having an Akaike weight close to 1, is the best model for the data. The log-log histogram in Figure 12 also helps us to rule out the possibility that the data set follows a power-law distribution. For this reason, we choose the log-normal distribution to set the parameter of the download frequencies of widgets.

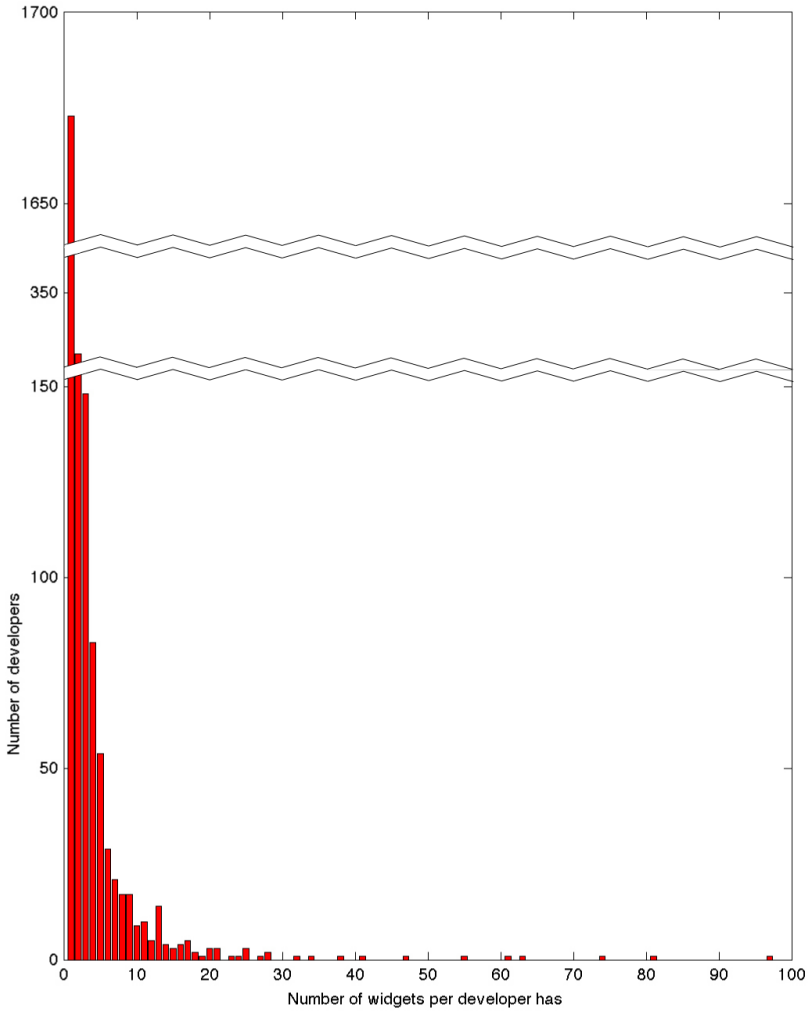

Figure 9: A histogram shows the number of developers (y-axis) as a function of the number of widgets each developer has uploaded(x-axis). A point *(x, y)* indicates how many developers (y-axis) have uploaded x widgets (x-axis).

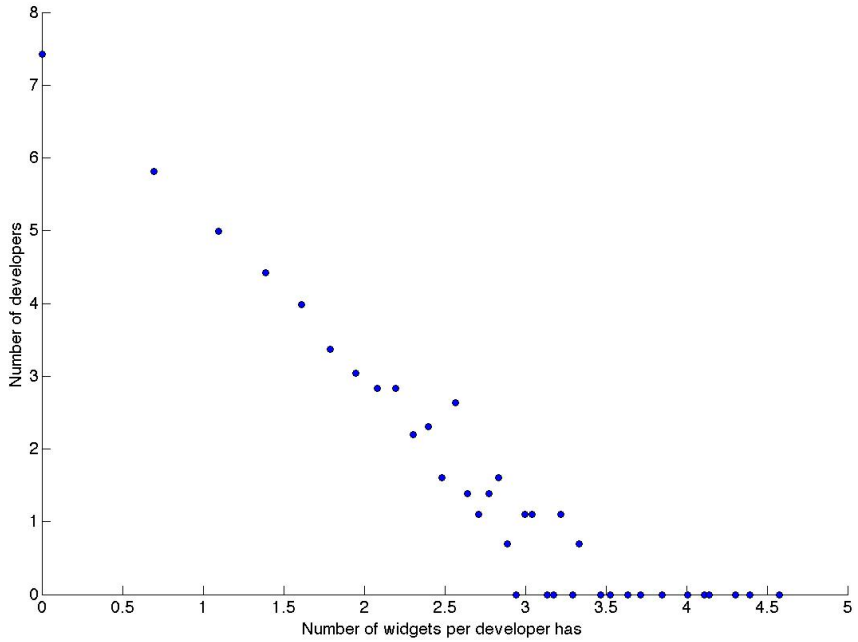

Figure 10: A log-log histogram of the number of developers (y-axis) as a function of the number of widgets per developer has (x-axis).

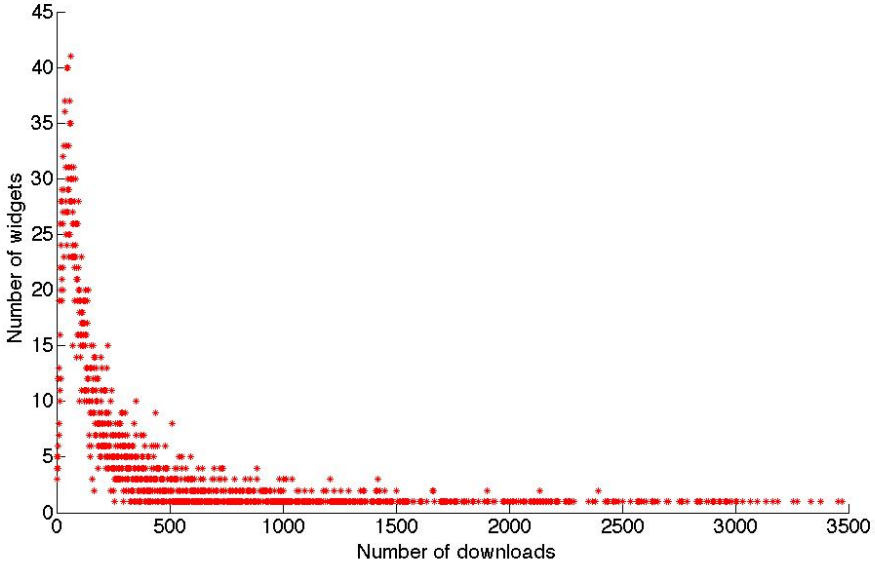

Figure 11: Histogram showing the number of widgets (y-axis) as a function of downloads (x-axis). A point *(x, y)* indicates how many widgets (y-axis) have been downloaded x-times.

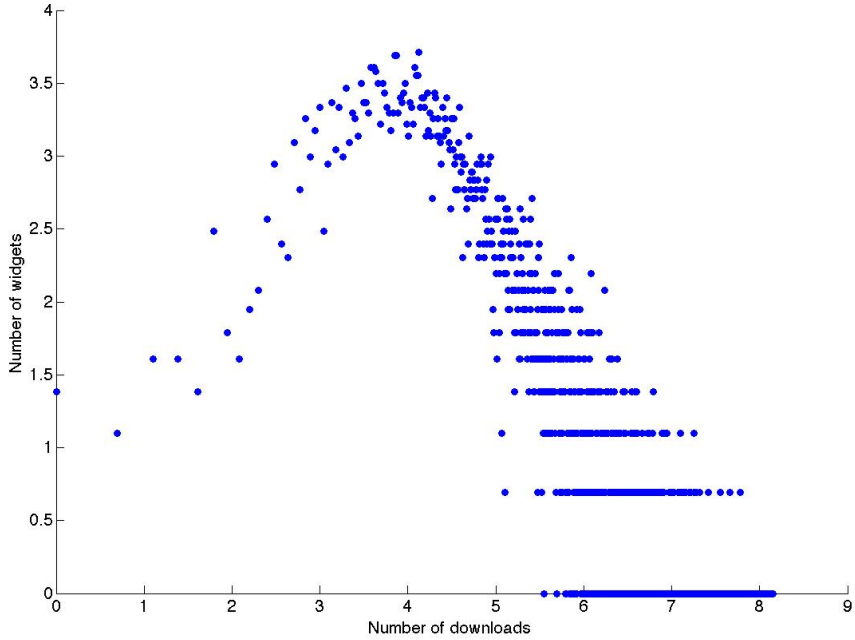

Figure 12: A log-log histogram of the number of widgets (y-axis) as a function of downloads (x-axis).

6 Comparison of Reputation Systems

The experiments focus on comparing the effectiveness of different reputation systems in the simulation environment proposed in the previous section. The experiment subjects cover all categories of reputation systems, as summarized in Table 2; see Section 3.4. In particular, we consider two simple systems, such as *eBay* from the Accumulative systems and *OnlyLast* from the Blurred systems, and four complex systems: *Average*, *BlurredSquared*, *Beta* and *Perseus*. Section 6.1 describes the experiment setup. Section 6.2 details the experiments and illustrates the results collected from these experiments. Section 6.3 summarizes the strengths and weaknesses of the studied reputation systems and provides recommendations based on the experiment results.

6.1 Experiment Setup

In order to empirically evaluate the performance of reputation systems and their vulnerabilities to attacks in widget sharing communities, we simulated different agent behavior models (Section 6.1.1), built up a model for acceptance behavior (Section 6.1.2) and defined the evaluation criteria (Section 6.1.3).

6.1.1 Behavior Models

In the experiments, we consider three types of developers: *Honest*, *Malicious*, and *Disturbing*.

1. **Honest.** Honest developers always create high quality widgets.
2. **Malicious.** Malicious developers upload good, neutral, or bad widgets by chance.
3. **Disturbing.** Disturbing developers attempt to build up a good reputation by acting honestly (i.e., uploading good widgets) and when having obtained a high reputation, they will start to create bad widgets until their reputation is badly damaged.

These behavior models, adapted from [SVB06], mainly concern developers' decision making in creating and uploading widgets; see Section 5.2.2. On the other hand, we considered four types of users.

1. **Honest.** Honest users only download good or neutral widgets that are uploaded by trustworthy developers. This type of users always rate the widget fairly according to the its outcome.

2. **Selfish.** Selfish users act similarly as the honest users, except that they never rate widgets.

3. **Spamming.** Spamming users occasionally download good, neutral or bad widgets, and always give negative rating to the widgets they used in order to undermine reputation systems.

4. **Misleading.** Misleading users accept all kinds of widgets, and always rate the downloaded widgets in an opposite way, i.e., give negative ratings to good widgets and positive ratings to bad widgets. By giving opposite ratings, they tend to mislead other users and undermine reputation systems.

These user models focus on users' decision making on downloading and rating widgets; see Section 5.2.1. With the exception of *Misleading*, these behavior models are adapted from [SVB06] and [Nur07].

6.1.2 Acceptance Model

It is assumed that the more popular a widget is, the higher probability that users choose the widget as a potential download candidate. We simplify users' widget selection policy and regard widgets' popularity as the unique standard for users to select widgets. Here, widget popularities are initialized using the log-normal distribution, which is determined from the empirical data; see Section 5.4.3.

It is assumed that once a widget has been selected, the trustworthiness of this widget and its developer influences users' download decisions. In our simulation, the trustworthiness of widgets and developers is determined by their ratings gained from the previous transactions. Based on the pre-defined rating thresholds (see Table 6), widgets and developers are categorized into three profiles: trustworthy, neutral and untrustworthy; see Table 7.

6.1.3 Evaluation Criteria

As the main evaluation criteria, we consider the *number of transactions* and the *false detection rate* of widgets.

Table 6: Rating threshold constants.

System	Threshold (T)			
	$T_{perfect}$	T_{good}	T_{bad}	T_{worst}
eBay	∞	10	-10	$-\infty$
Average	1	0.5	-0.5	-1
OnlyLast	1	0.9	-0.9	-1
BlurredSquared	$\frac{\pi^2}{6}$	0.6	0.2	$-\frac{\pi^2}{6}$
Beta	1	0.5	0.4	0
Perseus	1	0.7	0.3	0

Table 7: Profiles of widgets and developers defined based on the rating thresholds (T) in Table 6.

Profile	Range of T
Trustworthy	$[T_{good}, T_{perfect})$
Neutral	(T_{bad}, T_{good})
Untrustworthy	$(T_{worst}, T_{bad}]$

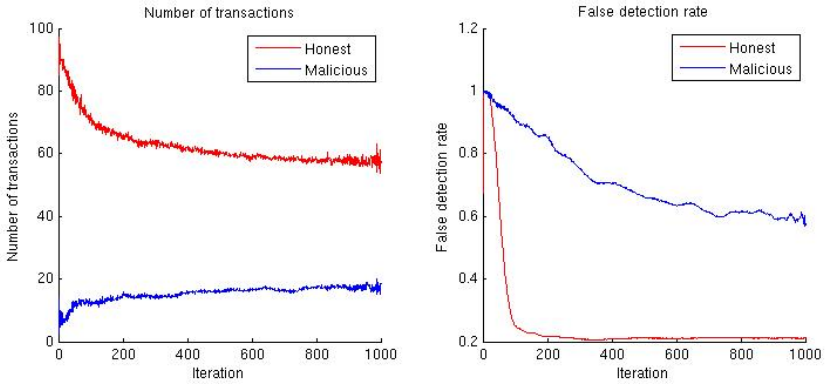

Figure 13: An example of evaluation criteria.

1. **Number of transactions.** The *number of transactions* of widgets measures the number of widgets of a certain type (i.e., honest or malicious) that have been downloaded.

2. **False Detection Rate.** A widget is *falsely detected* when its reputation is the exact opposite to its nature, e.g., an honest widget is *falsely detected* if it holds a reputation value which falls in the scale segment of the untrustworthy profile; see Section 6.1.2. Thus, the *false detection rate* measures the portion of each type of widgets that are incorrectly detected.

The *downloads* and the *false detection rate* are examined separately for widgets of each type (i.e., honest or malicious), while the *number of transactions* and the *false detection rate* are measured for developers and users in each behavior model; see Section 6.1.1. Schlosser et al. [SVB06] consider measurements that are based on the reputation values of the agents. In our case, our emphasis is on the downloads of honest widgets, because they ensure that the system remains operational. It is assumed that within widget sharing communities, if a reputation system works effectively, users will be able to distinguish honest widgets from malicious ones. As a result, honest widgets will become more and more popular and gain more downloads.

Figure 13 demonstrates an example of the evaluation criteria for widgets and developers. The horizontal axis depicts the number of iterations in all cases, whereas the vertical axis shows the value of the measured criterion.

6.2 Experiment Results

To examine the effectiveness of the reputation systems, two experiments were applied in the simulated widget sharing community described in Section 5. This section provides details to illustrate to which degree the different reputation systems can stand possible attacks from dishonest developers or users.

6.2.1 Experiment I: Varying Proportion of Dishonest Developers

The goal of this experiment was to investigate the maximum amount of dishonest developers at which the reputation systems are able to guarantee a majority of the users can download honest widgets. The number of users was determined using the normal distribution, with $\mu = 100$ and $\sigma^2 = 5$, to ensure the total number of users at the same level (≈ 100 users). In this experiment, it was assumed that all the users are honest. On the other hand, the number of developers was determined using the normal distribution ($\sigma^2 = 5$), and the proportions of honest and a certain type of dishonest developers, i.e., malicious or disturbing developers, were varied. The number of widgets each developer owns was simulated using the Pareto distribution to ensure that each developer has 4 widgets on average. Each developer and widget started with a reputation value of 0. For each reputation system, we ran 1000 iterations and repeated the same setup for 30 times.

In this experiment, the performance of each reputation system when coping with dishonest developers of different behavior model in each run were separately examined. The goal was to find the critical point where a reputation system fails to guarantee: (i) the downloads of honest widgets are more than that of malicious widgets; or (ii) the downloads of honest widgets falls below 40. For instance, given 100 users who are willing to download widgets, if the downloads of honest widgets is 30, with 10 more downloads comparing to that of malicious widgets, the reputation system still fails.

The results are summarized in Table 8. The values reported in the table are the maximum proportion of dishonest users at which the reputation system is able to remain resistant against this kind of dishonest users.

The results indicated that most systems were equally resistant to the attacks from disturbing and malicious developers. Within these examined systems, there was no significant difference in treating these two kinds of developers, because from the point of view of users, both the disturbing developers and the malicious ones

Table 8: Performance of different reputation systems. The values in the columns indicate the percentage of developers of the corresponding behavior type.

System	Malicious	Disturbing
eBay	60[a]	> 80
Average	70	> 80
OnlyLast	60	> 80
BlurredSquared	40	60
Beta	50	50
Perseus	50	> 80

[a]The system is resistant up to $x\%$ of this type dishonest developers. The rest are honest developers.

have a probability to provide malicious widgets and are the same kind of dishonest developers who may create malicious widgets. Since users do not interact with developers directly, they are not sensitive to the difference in behaviors between these two types of developers. In addition, the resistant level of these systems to disturbing developers was higher than that to malicious developers. Since the results of malicious developers corroborates the simulation results from Schlosser et al. [SVB06], the lower probability of disturbing developers to create malicious widgets can be one possible reason for this deviation.

6.2.2 Experiment II: Varying Proportion of Dishonest Users

This experiment separately examined the performance of each selected reputation system when coping with dishonest users of different behavior model. In each simulation run, the total number of users was kept at the same level (≈ 100 users), and the proportions of different types of users, i.e., spamming, selfish or misleading users, were varied. On the other hand, the number of developers was determined using the normal distribution ($\sigma^2 = 5$) and kept the distribution of developer behavior models fixed, i.e., $\mu_{Honest} = 40, \mu_{Malicious} = 5$ and $\mu_{Disturbing} = 5$. Thus, the total number of developers was around 50. The other experiment settings and critical point for judging if a system fails were the same as those presented in the previous section. Table 9 summarizes the results.

Table 9: Performance of different reputation systems. The values in the columns indicate the percentage of users of the corresponding behavior type.

System	Selfish	Spamming	Misleading
eBay	$> 80^a$	70	60
Average	> 80	60	60
OnlyLast	> 80	40	< 20
BlurredSquared	> 80	80	> 80
Beta	> 80	> 80	> 80
Perseus	> 80	80	70

[a]The system is resistant up to x% of this type dishonest users. The rest are honest users.

The results indicated that all the reputation systems were able to stand against dishonest users, except the OnlyLast system, which was weak against spamming users and vulnerable to misleading users. However, it is hard to determine whether a system is effective using the measurement *number of transactions* of widgets alone. To this end, the *false detection rate* of widgets was also considered in this experiment.

The false detection rate and the number of transactions of honest widgets in the reputation systems with certain amount of dishonest users are illustrated in Figures 14 - 19. Figures 14 and 15 show the results when 80% of all users are selfish. As the results indicated, the false detection rate of honest widgets in all reputation systems converged to 0 after 1000 iterations, except the eBay system, in which about 10%.honest widgets were regarded as malicious widgets. Figures 16 and 17 illustrate the results when 70% of all users are spamming. In this case, the Average system and the OnlyLast system failed to distinguish honest widgets from malicious ones. At the same time, the number of transactions of honest widgets decreased towards 0. Among all the reputation systems, the Beta system was the only one which can correctly detect honest widgets. Figures 18 and 19 demonstrate a similar situation when 30% of all users are misleading.

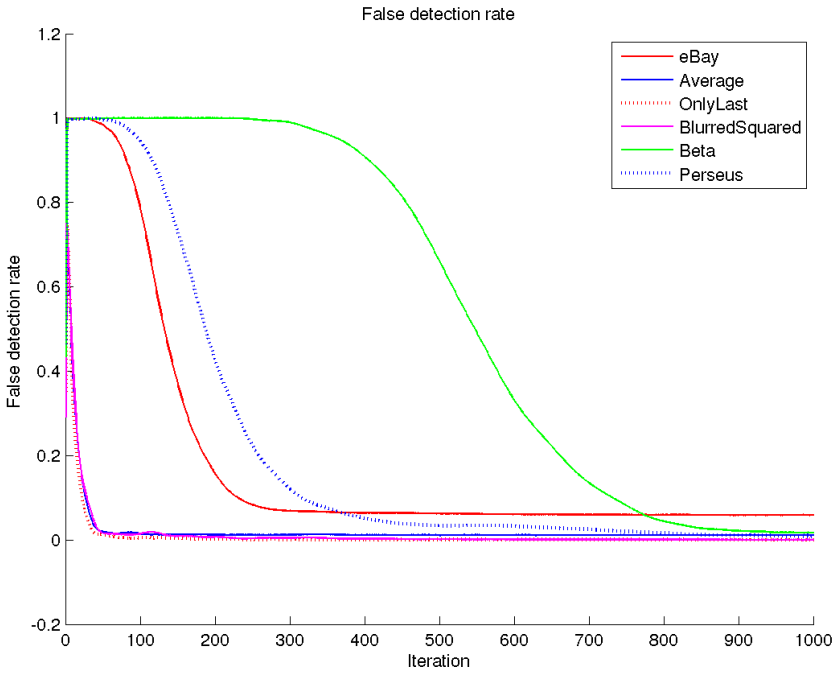

Figure 14: False detection rate of honest widgets in reputation systems with 80% SELFISH users.

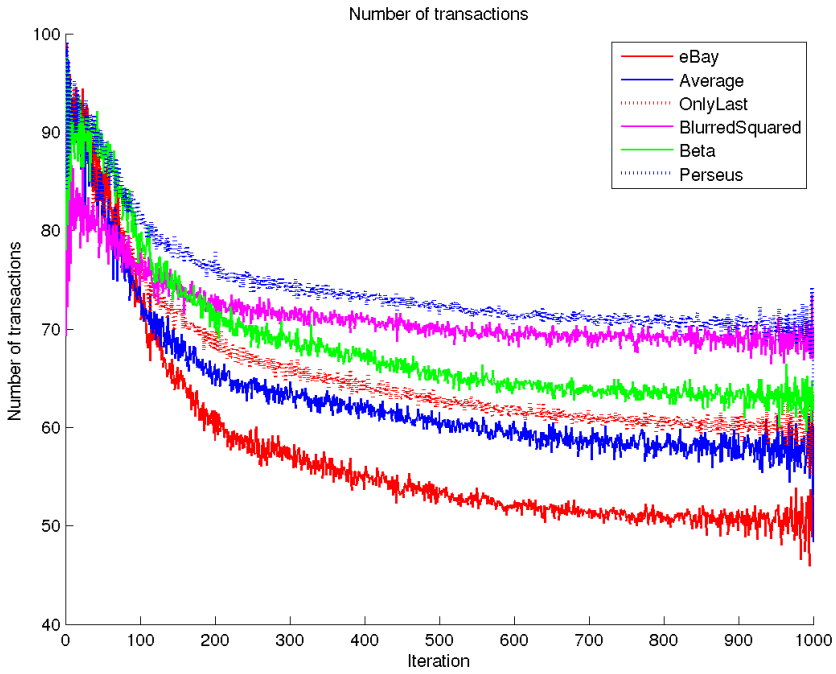

Figure 15: Number of transactions of honest widgets in reputation systems with 80% SELFISH users.

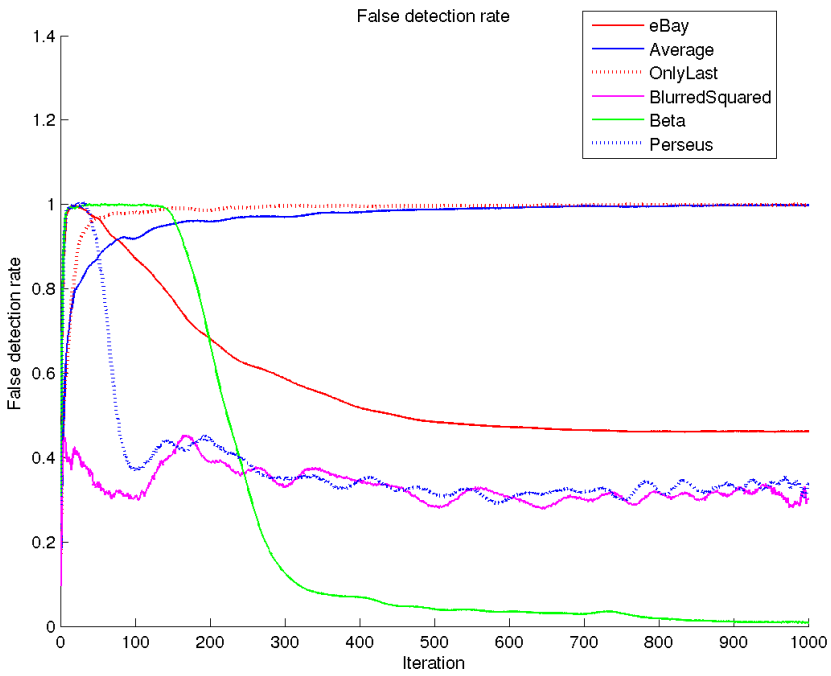

Figure 16: False detection rate of honest widgets in reputation systems with 70% SPAMMING users.

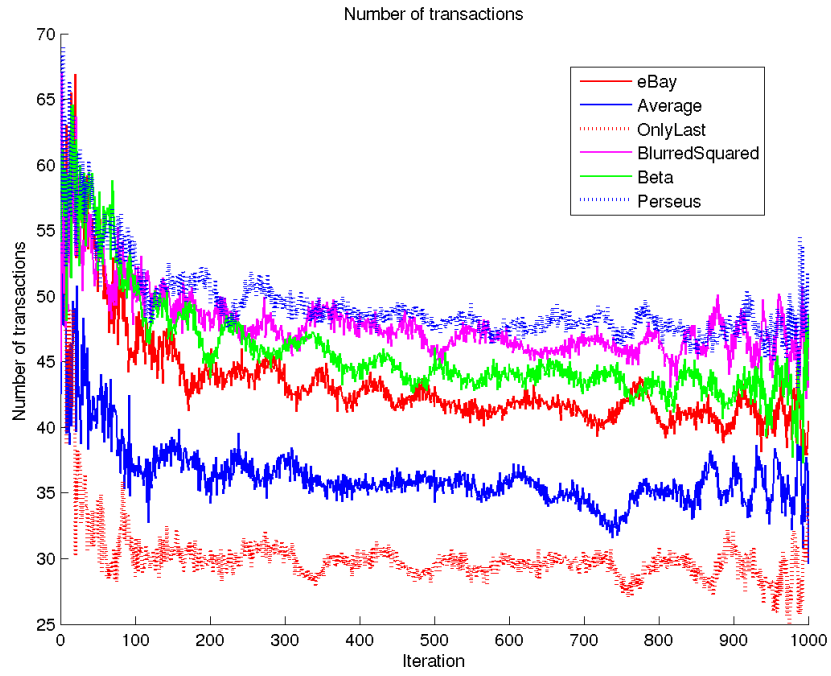

Figure 17: Number of transactions of honest widgets in reputation systems with 70% SPAMMING users.

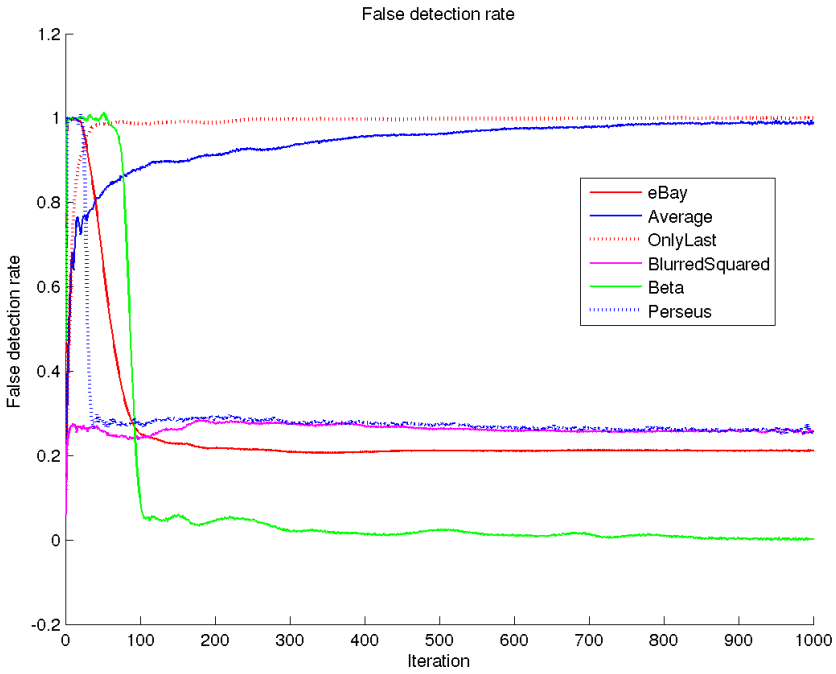

Figure 18: False detection rate of honest widgets in reputation systems with 30% MISLEADING users.

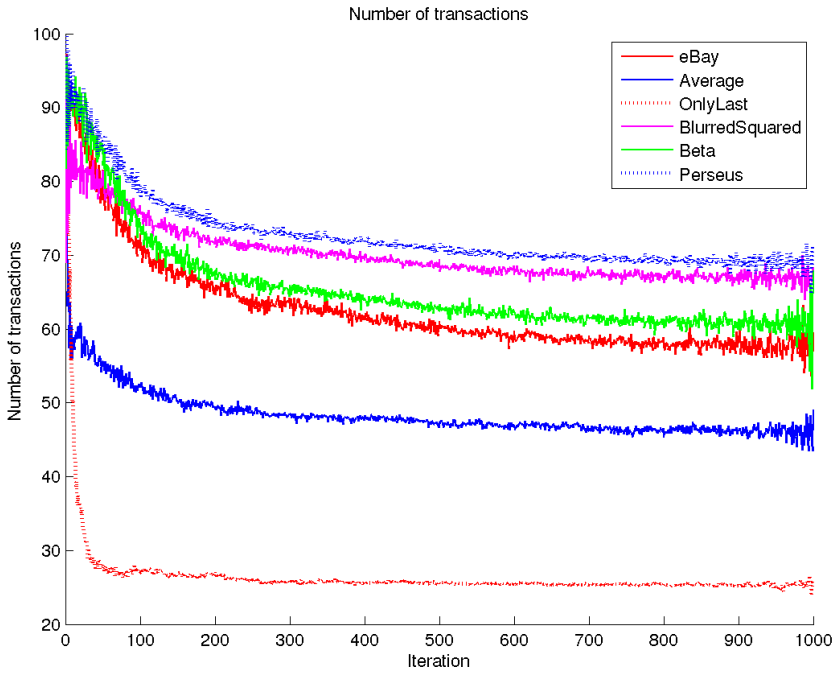

Figure 19: Number of transactions of honest widgets in reputation systems with 30% MISLEADING users.

6.3 Recommendations

Table 10 summarizes the evaluated reputation systems and their strengths as well as weaknesses; see Section 6.1.1 for the definitions of behavior models that are used in the table. It also provides some recommendations on selecting appropriate systems for widget sharing communities with different requirements.

Table 10: Summary of the reputation systems evaluated in the study. The definitions of the behavior models used in the table are given in Section 6.1.1.

System	Strengths	Weaknesses	Recommendation
eBay	• Relatively robust against dishonest users • Fast performance	• Suffers from a small number of user ratings	• Good for large-scale communities that require high performance reputation systems to handle a huge number of user ratings
Average	• Relatively robust against dishonest users • Fast performance	• Few downloads of honest widgets	
OnlyLast	• Fast performance	• Vulnerable to *spamming* and *misleading* users • Relatively few downloads of honest widgets	
BlurredSquared	• Resistant to dishonest users	• Sensitive to *malicious* developers	• Good for communities that want to provide users relatively reliable reputation information of widgets and maintain numerous downloads of honest widgets
Beta	• Good accuracy in detecting honest widgets • Resistant to dishonest users	• Slow on large datasets	• Good for small-scale communities that aim to provide users with reliable reputation information of widgets
Perseus	• Relatively more downloads of honest widgets • Resistant to dishonest users	• Relatively sensitive to *malicious* developers	• Good for communities that want to obtain a large number of users to download honest widgets

7 Conclusion

This study focused on the simulation of an online widget sharing community, consisting of users and developers where users download the widgets created by developers and provide feedback about the widgets after each transaction. A custom scenario was presented to describe the indirect interactions between users and developers, i.e., via widgets, and formed a simulation environment using statistical models determined by real data. In this thesis, six reputation systems were compared and evaluated within the simulated widget sharing environment to study their effectiveness in resisting the attacks from misbehaving developers and users, providing reliable reputation information of widgets to users, and encouraging users to download more good widgets. In more detail, through empirical evaluation, the experiment results show that:

1. The examined systems are able to handle attacks from dishonest developers if all the users behave honestly.
2. All of the examined reputation systems are susceptible to attacks from selfish users.
3. The OnlyLast system is vulnerable to spamming and misleading users.
4. The Beta system is able to provide reliable reputation information of widgets to users, which in turn can probably increase the downloads of honest widgets.
5. The Perseus system supports the highest downloads of honest widgets among the evaluated systems.

The following questions serve as suggestions for future work:

1. Is it feasible to integrate open multi-agent systems into this simulation so that we can form a more realistic environment which allows developers and users to freely join and leave the virtual widget sharing community?
2. Are there other reputation systems that can be included into this simulation? It should be interesting to integrate more systems into the simulation to better understand their strengths and weakness. This better understanding could provide the possibility to combine the good parts of different systems, and to design a system which works more effective to the widget sharing communities.

3. Is it possible to distinguish between honest users and misbehaving users in the community? How to treat them differently?

The first two questions can be regarded as possible extensions of the simulation work. The third one is more related to a cross work from both computer science and social aspects. It would be useful to investigate this problem in order to improve the trust among users and between users and developers. In summary, facing with the problems proposed above, both empirical and theoretical studies are needed in order to find out ideal solutions for each problem instance.

References

AM90 Ackerman, M. S. and Malone, T. W., Answer garden: a tool for grxowing organizational memory. *Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems*, New York, NY, USA, 1990, ACM, pages 31–39.

ARH00 Abdul-Rahman, A. and Hailes, S., Supporting trust in virtual communities. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, volume 1, 2000.

BA02 Burnham, K. P. and Anderson, D. R., *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2002.

BFK03 Barber, K. S., Fullam, K. and Kim, J., *Challenges for Trust, Fraud and Deception Research in Multi-agent Systems*, volume 2631. Springer, 2003.

BLB02 Buchegger, S. and Le Boudec, J.-Y., Performance analysis of the confidant protocol. *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc '02)*. ACM, 2002, pages 226–236.

BNF+08 Boström, F., Nurmi, P., Floréen, P., Liu, T., Oikarinen, T.-K., Vetek, A. and Boda, P., Capricorn - an intelligent user interface for mobile widgets. *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services (MobileHCI '08)*. ACM, 2008, pages 327–330.

BWW85 Beaver, W. L., Wasserman, K. and Whipp, B. J., Improved detection of lactate threshold during exercise using a log-log transformation. *Journal of Applied Physiology*, 59,6(1985), pages 1936–1940.

CBG02 Carter, J., Bitting, E. and Ghorbani, A. A., Reputation formalization for an information-sharing multi-agent system. *Computational Intelligence*, 18,4(2002), pages 515–534.

CP09 Caceres, M. and Priestley, M., Widgets 1.0: Requirements. Technical Report, The World Wide Web Consortium (W3C), 2009. URL `http://www.w3.org/TR/widgets-reqs/`.

Del03 Dellarocas, C., Efficiency and robustness of binary feedback: Mechanisms in trading environments with moral hazard. MIT Center for eBusiness Working Paper #170, 2003.

DF94 Drogoul, A. and Ferber, J., Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies. *Selected papers from the 4th European Workshop on on Modelling Autonomous Agents in a Multi-Agent World, Artificial Social Systems (MAAMAW '92)*, London, UK, 1994, Springer, pages 3–23.

EPW+07 Edwards, A. M., Phillips, R. A., Watkins, N. W., Freeman, M. P., Murphy, E. J., Afanasyev, V., Buldyerev, S. V., da Luz, M. G. E., Raposo, E. P., Stanley, H. E. and Viswanathan, G. M., Revisiting lévy flight search patterns of wandering albatrosses, bumblebees and dear. *Nature*, 449, pages 1044–1048.

FKM+05a Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K. S., Rosenschein, J. S., Vercouter, L. and Voss, M., A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS '05)*, New York, NY, USA, 2005, ACM, pages 512–518.

FKM+05b Fullam, K. K., Klos, T. B., Muller, G., Sabater, J., Topol, Z., Barber, K. S., Rosenschein, J. S. and Vercouter, L., A demonstration of the agent reputation and trust (art): testbed for experimentation and competition. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS '05)*, New York, NY, USA, 2005, ACM, pages 151–152.

FKM+05c Fullam, K. K., Klos, T. B., Muller, G., Sabater-Mir, J., Topol, Z., Barber, K. S., Rosenschein, J. and Vercouter, L., The agent reputation and trust (art) testbed architecture. *Proceedings 8th Workshop on Trust in Agent Societies*, volume 131 of *Frontiers in Artificial Intelligence and Applications*, 2005, pages 389–396.

FKM+06 Fullam, K. K., Klos, T. B., Muller, G., Sabater-Mir, J., Barber, S. K. and Vercouter, L., The agent reputation and trust (art) testbed. *Lecture*

Notes in Computer Science, volume 3986. Springer, 2006, pages 439–442.

FT91 Fudenberg, D. and Tirole, J., *Game Theory.* MIT Press, Cambridge, Massachusetts, 1991.

Gam88 Gambetta, D., Can we trust trust? *Trust: Making and Breaking Cooperative Relations.* Basil Blackwell, 1988, pages 213–237.

GJA03 Gupta, M., Judge, P. and Ammar, M., A reputation system for peer-to-peer networks. *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '03)*, New York, NY, USA, 2003, ACM, pages 144–152.

Gli99 Glickman, M. E., Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 48,3(1999), pages 377–394.

GMY04 Goldstein, M. L., Morris, S. A. and Yen, G. G., Problems with fitting to the power-law distribution. *The European Physical Journal B - Condensed Matter and Complex Systems*, 41, pages 255–258.

GS90 Gordon, J. and Shortliffe, E. H., The dempster-shafer theory of evidence. *Readings in uncertain reasoning*, San Francisco, CA, USA, 1990, Morgan Kaufmann Publishers Inc., pages 529–539.

Hew86 Hewitt, C., Offices are open systems. *ACM Trans. Inf. Syst.*, 4,3(1986), pages 271–287.

Hew91 Hewitt, C., Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47,1-3(1991), pages 79–106.

HJS06 Huynh, T. D., Jennings, N. R. and Shadbolt, N. R., An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13, pages 119–154.

HPZ06 Hu, N., Pavlou, P. A. and Zhang, J., Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication. *Proceedings of the 7th ACM conference on Electronic commerce (EC '06)*, New York, NY, USA, 2006, ACM, pages 324–330.

JF03 Jurca, R. and Faltings, B., Towards incentive-compatible reputation management. *Trust, Reputation, and Security: Theories and Practice*, 2003, pages 13–24.

JI02 Jøsang, A. and Ismail, R., The Beta reputation system. *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

JIB07 Jøsang, A., Ismail, R. and Boyd, C., A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43,2(2007), pages 618 – 644. Emerging Issues in Collaborative Commerce.

Mit04 Mitzenmacher, M., A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1, pages 226–251.

MM02 Mui, L. and Mohtashemi, M., A computational model of trust and reputation. *Proceedings of the 35th Hawaii International Conference on System Science (HICSS '02)*, 2002.

MMH02 Mui, L., Mohtashemi, M. and Halberstadt, A., Notions of reputation in multi-agents systems: a review. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS '02)*, New York, NY, USA, 2002, ACM, pages 280–287.

New05 Newman, M. E. J., Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46, page 323.

Nur07 Nurmi, P., Perseus – a personalized reputation system. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07)*, Washington, DC, USA, 2007, IEEE Computer Society, pages 798–804.

RK05 Ruohomaa, S. and Kutvonen, L., *Trust Management Survey*, volume 3477 of *Lecture Notes in Computer Science*. Springer, 2005.

RKZF00 Resnick, P., Kuwabara, K., Zeckhauser, R. and Friedman, E., Reputation systems. *Commun. ACM*, 43,12(2000), pages 45–48.

Rob95 Robert, C. P., Simulation of truncated normal variables. *Statistics and Computing*, 5,2(1995), pages 121–125.

RRH00 Ralston, A., Reilly, E. D. and Hemmendinger, D., *Encyclopedia of computer science (4th ed.)*. Grove's Dictionaries Inc., New York, NY, USA, 2000.

SFR00 Schillo, M., Funk, P. and Rovatsos, M., Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*.

SS02 Sabater, J. and Sierra, C., Reputation and social network analysis in multi-agent systems. *Proceedings of the first international joint conference on Autonomous agents and multiagent systems (AAMAS '02)*, New York, NY, USA, 2002, ACM, pages 475–482.

SS05 Sabater, J. and Sierra, C., Review on computational trust and reputation models. *Artificial Intelligence Review*, 24, pages 33–60.

STA05 Sakai, T., Terada, K. and Araragi, T., *Robust Online Reputation Mechanism by Stochastic Approximation*, volume 3394 of *Lecture Notes in Computer Science*. Springer, 2005.

SVB06 Schlosser, A., Voss, M. and Brückner, L., On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation*, 9,1(2006), page 1. URL `http://jasss.soc.surrey.ac.uk/9/1/4.html`.

Syc98 Sycara, K. P., Multiagent systems. *AI Magazine*, 19, pages 79–92.

VP06 Vandermeer, J. and Perfecto, I., A Keystone Mutualism Drives Pattern in a Power Function. *Science*, 311,5763(2006), pages 1000–1002.

WJ95 Wooldridge, M. and Jennings, N. R., Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10, pages 115–152.

XL03 Xiong, L. and Liu, L., A reputation-based trust model for peer-to-peer e-commerce communities. *IEEE International Conference on E-Commerce (EC '03)*, June 2003, pages 275–284.

YS02 Yu, B. and Singh, M. P., Distributed reputation management for electronic commerce. *Computational Intelligence*, 18,4(2002), pages 535–549.

YSS04 Yu, B., Singh, M. and Sycara, K., Developing trust in large-scale peer-to-peer systems. *IEEE First Symposium on Multi-Agent Security and Survivability*, 2004, pages 1–10.

ZMM00 Zacharia, G., Moukas, A. and Maes, P., Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems*, 29,4(2000), pages 371 – 388.