Towards a Trust and Reputation Framework for Social Web Platforms

Thao Nguyen[1,2,*], Luigi Liquori[1], Bruno Martin[2], and Karl Hanks

[1] Institut National de Recherche en Informatique et Automatique, France
[2] Université Nice Sophia Antipolis, France
{Thao.Nguyen,Luigi.Liquori}@inria.fr,
Bruno.Martin@unice.fr,
Karl.Hanks@cantab.net

Abstract. Trust and Reputation Systems (TRSs) represent a significant trend in decision support for Internet-based interactions. They help users to decide whom to trust and how much to trust a transaction. They are also an effective mechanism to encourage honesty and cooperation among users, resulting in healthy online markets or communities. The basic idea is to let parties rate each other so that new public knowledge can be created from personal experiences. The major difficulty in designing a reputation system is making it robust against malicious attacks. Our contribution in this paper is twofold. Firstly, we combine multiple research agendas into a holistic approach to building a robust TRS. Secondly, we focus on one TRS component which is the reputation computing engine and provide a novel investigation into an implementation of the engine proposed in [7].

1 Introduction

Information concerning the reputation of individuals has always been spread by word-of-mouth and has been used as an enabler of numerous economic and social activities. Especially now, with the development of technology and, in particular, the Internet, reputation information can be broadcast more easily and faster than ever before. Trust and Reputation Systems (TRSs) have gained the attention of many information and computer scientists since the early 2000s. TRSs have a wide range of applications and are domain specific. The multiple areas where they are applied, include social web platforms, e-commerce, peer-to-peer networks, sensor networks, ad-hoc network routing, and so on [5]. Among these, we are most interested in social web platforms. We observe that trust and reputation is used in many online systems, such as online auction and shopping websites, including eBay [1], where people buy and sell a broad variety of goods and services, and Amazon [2], which is a world famous online retailer. Online services with TRSs provide a better safety to their users. A good TRS can also create incentives for good behavior and penalize damaging actions. As noted by

[*] Corresponding author.

P. Herrero et al. (Eds.): OTM 2012 Workshops, LNCS 7567, pp. 13–22, 2012.

Resnick et al. [10], markets with the support of TRSs will be healthier, with a variety of prices and quality of service. TRSs are very important for an online community, with respect to the safety of participants, robustness of the network against malicious behavior and for fostering a healthy market.

From a functional point of view, a TRS can be split into three components, as justified in [9]. The first component gathers feedback on participants' past behavior from the transactions that they were involved in. This component includes storing feedback from users after each transaction they take part in. The second component computes reputation scores for participants through a Reputation Computing Engine (RCE), based on the gathered information. The third component processes the reputation scores, implementing appropriate reward and punishment policies if needed, and representing reputation scores in a way which gives as much support as possible to users' decision-making. A TRS can be centralized or distributed. In centralized TRSs, there is a central authority responsible for collecting ratings and computing reputation scores for users. Most of the TRSs currently on the Internet are centralized, for example the feedback system on eBay [1] and customer reviews on Amazon [2]. On the other hand, a distributed TRS has no central authority. Each user has to collect ratings and compute reputation scores for other users himself. Almost all proposed TRSs in the literature are distributed [7,9,5].

Some of the main unwanted behaviors of users that might appear in TRSs are: *free riding* (people are usually not willing to give feedback if they are not given an incentive to do so [10]), *untruthful rating* (users give incorrect feedback either because of malicious intent or because of unintended and uncontrolled variables), *colluding* (a group of users coordinate their behavior to inflate each other's reputation scores or bad-mouth other competitors. Colluding motives are only clear in a specific application), *whitewashing* (a user creates a new identity in the system to replace his old one when the reputation of the old one has gone bad), *milking reputation* (at first, a participant behaves correctly to get a high reputation and then turns bad to make a profit from their high reputation score). The milking reputation behavior is more harmful to social network services and e-commerce than to the others. More types of attacks can be found in [4,5].

In this section, we provide readers with a brief overview of TRSs with respect to their applications, components, classification and potential attacks. The rest of the paper is organized as follows. Sect. 2 introduces readers to our research methodology and gives an agenda for building a TRS which is robust against attacks. Sect. 3 details the implemented RCE. Sect. 4 reports the results from a thorough simulation on the engine. Sect. 5 and 6 discuss related work and our future areas of study respectively.

2 Trust and Reputation System Design Agenda

The design of a robust TRS is already partially addressed in several academic papers [5,4,3]. In this section, we aim to build on these studies and systematize the process of designing a TRS in general as in Fig. 1. First, we characterize the application system into which we want to integrate a TRS, and find and identify new elements of information which substitute for traditional signs of trust and reputation in the physical world [5]. Second, based on the characteristics of the application, we find suitable working mechanisms and processes for each component of the TRS, as already introduced in Sect. 1. This step should answer the following questions: "What kind of information do we need to collect and how?", "How should the reputation scores be computed using the collected information?", and "How should they be represented and processed to lead users to a correct decision?". To answer the first question, which corresponds to the information gathering component, we should take advantage of information technology to collect the vast amounts of necessary data [5]. According to [5], a RCE should meet these criteria: *accuracy* for long-term performance (distinguishing a newcomer with unknown quality from a low-quality participant who has stayed in the system for a long time), *weighting* towards recent behavior, *smoothness* (adding any single rating should not change the score significantly), and *robustness* against attacks. The work in [3] is an effective guide for social web applications. It is applicable directly to the tasks of designing the information gathering and decision support components. Third, we study the tentative design obtained after the second step in the presence of selfish behaviors. During the third step, we can repeatedly return to Step 2 whenever appropriate until the system reaches a desired performance. The fourth step will refine the TRS and make it more robust against malicious attacks, some of which are listed in Sect. 1. If a modification is made, we should return to Step 2 and check all the conditions in steps 2 and 3 before accepting the modification (Fig. 1).

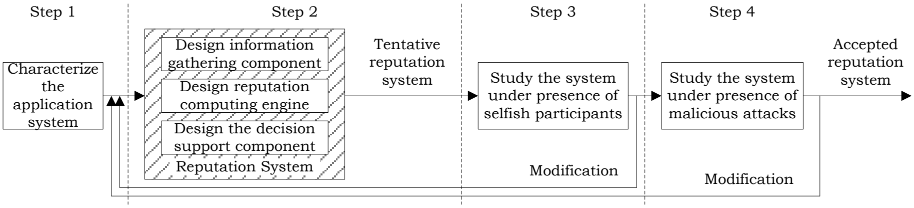
Step 1
Step 2
Step 3
Step 4
Characterize the application system
Design information gathering component
Design reputation computing engine
Design the decision support component
Reputation System
Tentative reputation system
Study the system under presence of selfish participants
Study the system under presence of malicious attacks
Accepted reputation system
Modification
Modification

Fig. 1. Process of designing a robust trust and reputation system

In different applications, there are different kinds of available information and activities, hence different ways of computing reputation scores and different types of attacks. Accordingly, designing a TRS must be put in the specific context of an application. Most of the challenges for a TRS are induced by selfish and malicious behaviors. The problems arising from malicious behaviors are usually more complicated and difficult to cope with than those caused by normal selfish users. Therefore, the logic of our methodology is to first design a TRS that works for a community where all the members are obedient, and then increase the sophistication of the system to cope with selfish members and finally with malicious ones. After having a TRS which is robust against some kinds of attacks, we can continue testing its robustness, using the approaches proposed in [4]. These are: implementing the system in reality, performing a theoretical test by third parties so that the evaluation is more credible, and defining a comprehensive set of robustness evaluation methods and criteria.

3 Reputation Computing Engine

Among the three components of a TRS, information gathering is most dependent on the application system, followed by the decision support component and then by the RCE. Accordingly, the next step in our research will be building a robust RCE, which will be as general as possible so that the engine is applicable to a variety of applications. In the following part of this section, we will elaborate on our assumptions, concepts and the implementation of a preliminary computing engine which is a specification and a simplification of the framework proposed in [5].

3.1 Assumptions and Notations

There is a large group of systems where the transactions are bilateral. In these systems, for each transaction there are two parties that we call *consumer* and *provider*. The consumer is the one who requests the service, while the provider is the one who is capable of providing the service. When we add a TRS to this kind of system, a user can have an additional role as a rater who has interacted with the provider before and therefore has an opinion about the provider's service. When a consumer needs a service, he collects ratings on the candidate providers and computes reputation scores for them. The consumer then ranks the providers according to their reputation scores and chooses one of the top ranked providers to interact with.

Without losing generality, we consider the RCE within the context of a service having one criterion to be judged. An example of service with multiple criteria is that provided by an eBay user [1]. As of May 2012, this service has been judged by four criteria, including "Item as described", "Communication", "Shipping time", and "Shipping and handling charges". The single criterion in our example is called "Quality of Service" (QoS), whose value is normalized to the interval $[0, 1]$, where 1 is the optimal value. Correspondingly, the rating value is in the range $[0, 1]$ and the optimal value is also 1. The following are the main variables that a consumer will use for his computation. They are private information and accessible only to the owner.

Rater Credibility (C_r): reputation of a user in giving accurate feedback. It reflects how much the consumer should trust the rater r's ratings. The value range is $[0, 1]$ and it is initiated at 0.5.

Usefulness Factor (U_r): $U_r = N_{useful}/S$, where N_{useful} is the number of times that the rater r submits useful ratings, S is the total number of submissions. After a transaction, if the difference between a rating and the outcome observed by the consumer is under a predefined threshold then the rating is useful. We set the threshold at 0.2 and the initial value of U_r at 0.5.

Personal Evaluation (E_{p}): the consumer's first-hand experience with the provider p. $E_{\mathrm{p}} \in [0,1]$. We set E_{p} as the experience of the last transaction with the provider. It might not be available if the consumer has never done any transactions with this provider.

Previously Assessed Reputation Score (A): the last computed reputation score of a provider when the consumer interacted with him. If the consumer has never interacted with this provider, then A will be initiated as 0.5.

3.2 Adjusting Raters' Credibility

After collecting ratings on a provider's service, the consumer will adjust raters' credibility which will be used to calculate weights for the ratings. The two main criteria which adjust a rater's credibility are the consistency of his rating to other raters and to the previous reputation score A of the provider. We use a modification of the K-means clustering algorithm in [6] to find the majority rating value among the raters. The main idea of this algorithm is to divide the rating set into clusters, so that similar ratings are grouped into the same cluster, while different ones are separated into different clusters. The most crowded cluster is then labeled as the majority and its mean is the majority rating M. The values used to decide if ratings belong to the same or different clusters are affected by coarsening and refinement distances, C and R respectively: $0 \leq C \leq R \leq 1$. The three parameters of the algorithm are C, R, and the initial number of clusters. After having M, the consumer computes factor $M_r^f \in [0,1]$, which has the effect of changing rater r's credibility, due to the closeness of its rating R_r to M.

$$M_r^f = \begin{cases} 1 - \dfrac{|R_r - M|}{\sigma_M} & \text{if } |R_r - M| < \sigma_M \\[2ex] 1 - \dfrac{\sigma_M}{|R_r - M|} & \text{otherwise} \end{cases}$$

where σ_M is the standard deviation of the received rating set, i.e.:

$$\sigma_M = \sqrt{\frac{\sum_{r=1}^{N_R} R_r^2}{N_R} - (\sum_{r=1}^{N_R} \frac{R_r}{N_R})^2}$$

with N_R is the total number of collected ratings. Factor $A^f = 1$ has an effect on the change of a rater's credibility, due to the closeness of its rating to A. We denote $\aleph = C_r \times (1 - |R_r - M|)$, and ρ the consumer's pessimism factor which has a suggested minimum value of 2 ($\rho \geq 2$). The credibility C_r of rater r is adjusted as follows:

1. If rating R_r is similar to both M and A, i.e., ($|R_r - M| < 0.1$) and ($|R_r - A| < 0.1$), then: $C_r = \min(1, C_r + \aleph \times \dfrac{M_r^f + A^f}{\rho})$.
2. If ($|R_r - M| < 0.1$) and ($|R_r - A| \geq 0.1$), then C_r is still increased, but less than the first case: $C_r = \min(1, C_r + \aleph \times \dfrac{M_r^f}{\rho})$.

3. If ($|R_r - M| \geq 0.1$) and ($|R_r - A| < 0.1$), then C_r is decreased a little bit: $C_r = \max(0, C_r - \aleph \times \dfrac{A^f}{\rho})$.
4. If ($|R_r - M| \geq 0.1$) and ($|R_r - A| \geq 0.1$), then C_r is decreased the most: $C_r = \max(0, C_r - \aleph \times \dfrac{M_r^f + A^f}{\rho})$.

According to the above formulas, a pessimistic consumer with high ρ will increase C_r slowly for a rating consistent with M and A. Finally, $C_r = C_r \times U_r$.

3.3 Computing Assessed Reputation Score

To prepare weights for collected ratings, in addition to the credibility of raters, we also need to calculate *temporal factors* (f^t) for ratings and E_p. The reason for using f^t is to give more weight to the more recent information. Depending on the characteristics of the service and the discretion of the designers, f^t can be calculated in different ways. We propose one example here. Ratings and E_p are arranged into chronological order and then a f_r^t corresponding to each rating R_r or f_E^t for E_p is calculated, which is the inverse of the number of ratings (S) counted between R_r and the latest rating inclusively: $f_r^t = 1/S$. Then the assessed reputation score of provider p is computed as in the following formula:

$$Rep_p = \frac{(\sum_{r=1}^{N_R}(R_r \times C_r \times f_r^t) + E_\mathrm{p} \times f_\mathsf{E}^t)}{(\sum_{r=1}^{N_R} C_r + 1)}$$

Where N_R is the total number of collected ratings and E_p is the consumer's first-hand experience (if available) with the provider.

4 Simulation Results

We have implemented the RCE described above and tested it under different user behaviors. The details of our simulation in Java language are the following.

4.1 Simulation Settings

We set up a population of N_u users, providing the same service, and undertaking N_t transactions. In each transaction, a random consumer is assigned to request the service. Other users will then be candidate providers for this request. When a user plays the role of a consumer, his behavior is modeled in *raterType* attribute. Three types of raters include HONEST, DISHONEST and COLLUSIVE. HONEST raters share their personal experience honestly, i.e. $R_r = E_\mathrm{p}$. DISHONEST raters provide ratings 0.5 different from their true estimation, i.e. $R_r = E_\mathrm{p} \pm 0.5$. COLLUSIVE raters give the highest ratings ($R_r = 1$) to users in their collusion and the lowest ratings ($R_r = 0$) to the rest. Similarly, when a user acts as a provider, he can be one of the following types of providers: GOOD, NORMAL, BAD, or GOODTURNBAD. This type is denoted in *providerType* attribute.

The QoS of the service provided by a BAD, NORMAL, or GOOD provider has a value in the interval $(0, 0.4]$, $(0.4, 0.7]$, or $(0.7, 1]$ respectively. A GOODTURNBAD provider will change the QoS of his service when 50% of N_t transactions have been done in the simulation. To get a transaction done, a consumer obtains a list of providers, computes reputation scores for them, chooses a provider to perform the transaction, updates his private information, and publishes his rating for the provider. The quality of service that the consumer will experience depends on the *providerType* of the chosen provider. The difference between the consumer's rating for the provider and his observation depends on the consumer's *raterType*. In our simulation, *providerType* and *raterType* of a user are independent.

For each user, we have two measures: p_p which is the percentage of N_t transactions, in which the user has performed as the provider; and $a_d = \overline{|Rep_p - E_{\mathrm{p}}|}$ which is an average of absolute difference between the Rep_p before a transaction and the E_{p} after the transaction. Apparently, p_p should be proportional to the user's QoS and can be referred to as the user's "Market Share". a_d reflects the correctness of the system in assessing a provider's QoS and is captured when a user plays the role of a consumer. We repeat a simulation at least five times before taking the average values to analyze. A simulation run is denoted:

$$\textbf{\textit{Simulation(}}N_u\textbf{\textit{,}}\ N_t\textbf{\textit{,}}\ \%G\textbf{\textit{,}}\ \%N\textbf{\textit{,}}\ \%B\textbf{\textit{,}}\ \%GTB\textbf{\textit{,}}\ \%H\textbf{\textit{,}}\ \%D\textbf{\textit{,}}\ \%C\textbf{\textit{,}}\ \%dataLost\textbf{\textit{).}}$$

Where $\%G$, $\%N$, $\%B$, $\%GTB$ are the percentage of GOOD, NORMAL, BAD, and GOODTURNBAD providers in the population respectively, so that $\%G + \%N + \%B + \%GTB = 100\%$. $\%H$, $\%D$, $\%C$ are the percentage of HONEST, DISHONEST, and COLLUSIVE raters respectively, so that $\%H + \%D + \%C = 100\%$. $\%dataLost$ is the percentage of ratings on a specific provider which are not available for the consumer at the moment he computes the reputation score for the provider.

4.2 User Decision Simulation

We observe that, in reality, a user might not always choose a provider with the highest reputation score. This is due to the complexity of human decision-making, which is based not only on reputation and personal experience but also on many other factors, such as *aversion*, *bias*, *antecedents* and *mood*. For the purpose of examining a TRS, we model a user's decision-making in the following selection strategy. The consumer ranks candidate providers based on their reputation scores and makes a cut-off, removing those having scores lower than $(Top - 0.5)$, where Top is the score of the first ranked provider. He then uses a Gaussian distribution having standard deviation $\sigma = \sqrt{N_{Size}}$ and mean $a = 0$ to calculate Gaussian random values (G_p)s for providers in the short list of size N_{Size}. Accordingly, the G_p of a provider depends on N_{Size} and its rank in the list but not on its absolute reputation score Rep_p. Finally, the consumer opts for a provider randomly, with a probability proportional to the provider's G_p. The intuition of this strategy is that the providers with higher Rep_p, therefore higher rank, will have higher chance to be chosen.

4.3 Simulation Scenarios and Analysis

In this section, we are going to apply the methodology mentioned in Sect. 2 to study the implemented RCE. We examine the engine to see firstly, if it works correctly when all users are obedient, secondly if it is robust against selfish users and thirdly, if it is robust against malicious users.
Obedient Users. We consider obedient users to be those who provide a service correctly, as stated, to the best of their ability, and share their personal experience honestly with the community. Applying this concept to our simulation model, they are GOOD, NORMAL, or BAD providers and HONEST raters. Simulation on obedient users with parameters:

Simulation(200, 10000, 10, 20, 70, 0, 100, 0, 0, 0).

shows that BAD providers are avoided always.
Selfish Users. A selfish behavior in generic TRSs can be named as *free riding*. The consumer does not give feedback to the system after a transaction. We simulated that behavior in an approximate manner using the parameter $\%dataLost$:

Simulation(200, 10000, 10, 20, 70, 0, 100, 0, 0, 60).

The result obtained is almost the same as the one when all the users are obedient. It proves that the engine still functions even when 60% of the ratings which are supposed to be supplied are not available.
Malicious Raters. As raters, malicious users can be categorized as DISHONEST or COLLUSIVE. DISHONEST raters act individually, while COLLUSIVE ones act in groups. It is difficult to identify COLLUSIVE users, especially when they form a large group. The simulation with the presence of DISHONEST raters:

Simulation(200, 10000, 10, 20, 70, 0, 30, 70, 0, 0).

shows that the error in computing (Rep_p)s for BAD providers is 0.39 on average. And this error gives them a chance to acquire 13% × N_t transactions. Second, we run a simulation with the presence of COLLUSIVE raters, where 60% of users collude as a group against the rest:

Simulation(200, 10000, 10, 20, 70, 0, 40, 0, 60, 0).

The error in computing (Rep_p)s are now increased for all types of providers (Fig. 2). Especially, for BAD users, the error reaches 0.57, which is quite high. However, the market share of these BAD providers is only 7%, which is acceptable compared to their population percentage of 60%.
Malicious Providers. A provider displaying malicious behavior and *milking reputation* can be modeled by GOODTURNBAD users:

Simulation(200, 10000, 10, 10, 70, 10, 100, 0, 0, 0).

We observe that when the users change the QoS of their service from high to low, they continue to get high reputation scores and have a high chance of being

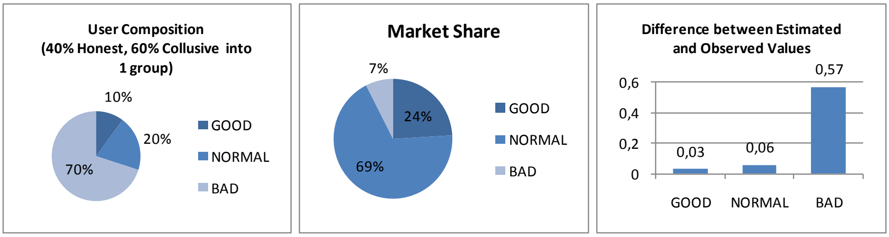

Fig. 2. Simulation results for a scenario with COLLUSIVE raters forming a group

selected for many transactions later. Even a consumer who has experienced the bad service of a GOODTURNBAD provider, can still choose the provider again. From these results, we conclude that the current engine is robust against a population displaying dishonest behavior of up to 70% and colluding behavior of up to 60% of the population, but still vulnerable to *milking reputation* attack.

5 Related Work

After a thorough survey of the current research, we find the model proposed in [7] the most interesting. The service oriented environment analyzed in the paper fits into the bilateral transaction environment we are aiming at. The experimental results that are provided are appealing. However, the disadvantage of this model is that it is complicated to implement. Furthermore, it is not clear which specific formulas and parameter values were used by the authors to get the results presented in the paper. In their proposed framework, a number of formulas are left open to readers. Such formulas include the one for updating personal evaluations and the one for aggregating the provider's assessed reputations at previous time instances. Some other points that are unclear in the paper, are the thresholds used to estimate if a rating is useful, to decide if a rating is similar to the majority rating and if it is similar to previously assessed reputations. In our opinion, the variance of these formulas and thresholds all affect the precision of assessed reputations and this concern has forced us to re-implement their model but using a simpler form. From a technical point of view, we have adopted the credibility update algorithm from the RateWeb framework [7]. Then we apply our methodology to study the robustness of the implemented engine. For the simulation, we propose a new measure which is the percentage of market share that a user gains. In terms of the accuracy of the reputation values, the results in our simulation are not as good as in [7] due to the modification we made to the engine and the simulation/experimental settings. The critical difference between the experimental settings in [7] and our simulation settings is the existence of bootstrapping. We assign a neutral default value to newcomers (initiating A, C_r, and U_r to 0.5) as a simple bootstrapping mechanism integrated into the computing engine and let every user start from scratch. On the other hand, [7] assumes that the bootstrapping has been done and that the system is running with correct reputation scores and credibilities.

6 Conclusions and Future Work

In this paper, we have presented our preliminary work on building a trust and reputation framework for social web platforms which will be robust against attacks. We propose a research methodology which can be used to study the robustness of many TRSs, and also implement a model which is a simplified and modified version of the RateWeb engine [7]. Since the experimental results in [7] were no longer correct for the implemented engine, we applied our methodology to study it. The results of the simulation showed a flaw in the engine, which is vulnerable to *milking reputation* attack. For that reason, we have decided to design a new RCE. During the course of studying the implemented engine, we realized that we needed a common tool and measuring system to compare the performance of multiple engines. Unfortunately, there is very little work in the literature on simulation tools and TRS performance measures. We have found only one implemented simulator for TRSs in sensor networks [8], whose measuring system is not applicable to our context. Therefore, another branch of our research in the future will be building a simulator with measures suitable for social web applications, as we have introduced partly in Sect. 4. We are interested not only in the accuracy of the TRSs, but also in how they shape the community.

References

1. http://www.ebay.com/
2. http://www.amazon.com/
3. Dellarocas, C.: Online reputation systems: How to design one that does what you need. MIT Sloan Management Review 51(3) (spring 2010)
4. Josang, A., Golbeck, J.: Challenges for robust trust and reputation systems. In: Proceedings of the 5th International Workshop on Security and Trust Management, Saint Malo, France (September 2009)
5. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43(2), 618–644 (2007)
6. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
7. Malik, Z., Bouguettaya, A.: Rateweb: Reputation assessment for trust establishment among web services. The International Journal on Very Large Data Bases 18(4), 885–911 (2009)
8. Marmol, F.G., Perez, G.M.: Trmsim-wsn, trust and reputation models simulator for wireless sensor networks. In: IEEE International Conference on Communications (IEEE ICC 2009), Communication and Information Systems Security Symposium (June 2009)
9. Marti, S.: Trust and Reputation in Peer-to-Peer Networks. PhD thesis, Stanford University. Stanford InfoLab (May 2005)
10. Resnick, P., Zeckhauser, R., Friedman, E., Kuwabara, K.: Reputation systems. Communications of the ACM, pp. 45–48 (December 2000)