Sänger *et al. Journal of Trust Management* (2015) 2:5
DOI 10.1186/s40493-015-0015-3

Journal of Trust Management
a SpringerOpen Journal

RESEARCH Open Access

Reusable components for online reputation systems

Johannes Sänger*, Christian Richthammer and Günther Pernul

*Correspondence:
johannes.saenger@wiwi.uni-regensburg.de
University of Regensburg,
Universitätsstraße 31, 93053
Regensburg, Germany

Abstract
Reputation systems have been extensively explored in various disciplines and application areas. A problem in this context is that the computation engines applied by most reputation systems available are designed from scratch and rarely consider well established concepts and achievements made by others. Thus, approved models and promising approaches may get lost in the shuffle. In this work, we aim to foster reuse in respect of trust and reputation systems by providing a hierarchical component taxonomy of computation engines which serves as a natural framework for the design of new reputation systems. In order to assist the design process we, furthermore, provide a component repository that contains design knowledge on both a conceptual and an implementation level. To evaluate our approach we conduct a descriptive scenario-based analysis which shows that it has an obvious utility from a practical point of view. Matching the identified components and the properties of trust introduced in literature, we finally show which properties of trust are widely covered by common models and which aspects have only rarely been considered so far.

Keywords: Trust; Reputation; Reusability; Trust pattern

Introduction

In the last decade, trust and reputation have been extensively explored in various disciplines and application areas. Thereby, a wide range of metrics and computation methods for reputation-based trust has been proposed. While most common systems have been introduced in e-commerce, such as eBay's reputation system [1] that allows to rate sellers and buyers, considerable research has also been done in the context of peer-to-peer networks, mobile ad hoc networks, social networks or ensuring data accuracy, relevance and quality in several environments [2]. Computation methods applied range from simple arithmetic over statistical approaches up to graph-based models involving multiple factors such as context information, propagation or personal preferences. A general problem is that most of the newly introduced trust and reputation models use computation methods that are designed from scratch and rely on one novel idea which could lead to better solutions [3]. Only a few authors build on proposals of others. Therefore, approved models and promising approaches may get lost in the shuffle.

In this work, we aim to encourage reuse in the development of reputation systems by providing a framework for creating reputation systems based on reusable components. Design approaches for reuse have been given much attention in the software engineering community. The research in trust and reputation systems could also profit from benefits like effective use of specialists, accelerated development and increased reliability. Toward this goal, we propose a *hierarchical taxonomy* for components of computation engines used in reputation systems. Thereto, we decompose the computation phase of common reputation models to derive single building blocks. The classification based on their functions serves as a natural framework for the design of new reputation systems. Moreover, we set up a *component repository* containing artifacts on both a conceptual and an implementation level to facilitate the reuse of the identified components. On the conceptual level, we describe each building block as a design pattern-like solution. On the implementation level, we provide already implemented components by means of web services.

The rest of this paper is based on the design science research paradigm involving the guidelines for conducting design science research by Hevner et al. [4] and organized as follows: Firstly, we give an overview of the general problem context as well as the relevance and motivation of our work. Thereby, we identify the research gap and define the objectives of our research. In the following section, we introduce our hierarchical component taxonomy of computation engines used in reputation systems. After that, we point out how our component repository is conceptually designed and implemented. Subsequently, we carry out a descriptive scenario-based analysis of our approach. At the same time, we match all components identified with the properties of trust introduced in literature. We show which properties of trust are widely covered by common models and which aspects have only rarely been considered so far. Finally, we summarize the contribution and name our plans for future work.

Problem context and motivation

With the success of the Internet and the increasing distribution and connectivity, trust and reputation systems have become important artifacts to support decision making in network environments. To impart a common understanding, we firstly provide a definition of the notion of trust. At the same time, we explain the properties of trust that are important with regard to this work. Then, we point out how trust can be established applying computational trust models. Focusing on reputation-based trust, we explain how and why the research in reputation models could profit from reuse. Thereby, we identify the research gap and define the objectives of this work.

The notion of trust and its properties

The notion of trust is a topic that has been discussed in research for decades. Although it has been intensively examined in various fields, it still lacks a uniform and generally accepted definition. Reasons for this circumstance are the multifaceted terms trust is associated with like credibility, reliability or confidence as well as the multidimensionality of trust as an abstract concept that has a cognitive, an emotional and a behavioral dimension. As pointed out by [5], trust has been described as being structural in nature by sociologists while psychologists viewed trust as an interpersonal phenomenon. Economists, however, interpreted trust as a rational choice mechanism. The definition often cited in literature regarding trust and reputation online that is referred to as *reliability trust* was proposed by Gambetta in 1988 [6]: "*Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another* *agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action."*

Multiple authors furthermore include security and risk which can lead to more complex definitions. Anyway, it is generally agreed that trust is multifaceted and dependent on a variety of factors. Moreover, there are several properties of trust described in literature (see Table 1). These properties are important with respect to this work because they form the basis for many applied computation techniques in trust and reputation systems described in Section 'Hierarchical component taxonomy'. Reusable components could extend current models by the ability to gradually include these properties.

Reputation-based trust

In recent years, several trust models have been developed to establish trust. Thereby, two common ways can be distinguished, namely policy-based and reputation-based trust

Table 1 Overview of properties of trust described in literature [14,41-46]

Dynamic	Trust can increase or decrease through gathering new experiences. Moreover, trust is said to decay with time (time-based aging [45]). Because of these characteristics, trust values strongly depend on the time they are determined. The greater importance of new experiences compared to old experiences has been widely studied and considered in many trust models such as [32,47] or [30].
Context-dependent	Trust is bound to a specific context. For example, Alice trusts Bob as her doctor. However, she might not trust him as a cook to prepare a delicious meal for her.
Multi-faceted	Even in the same context, a trust value may not reflect all aspects of this context [43]. For example, a customer may trust a particular restaurant for its quality of food but not for its quality of service. The overall trust on this restaurant depends on the combination of the amount of trust in the specific aspects.
Propagative	One property of trust made use of in several models is its propagativity. If Alice trusts Bob, who in turn trusts Claire, Alice can derive trust on Claire from the relationships between her and Bob as well as between Bob and Claire. Because of this propagative nature, it is possible to create trust chains passing trust from one agent to another agent. As clarified by Christianson and Harbison [48], trust is not automatically transitive although trust transitivity was assumed proven for a long time. If Alice trusts Bob, who in turn trusts Claire, it does not inherently mean that Alice trusts Claire. It follows from the foregoing that transitivity implies propagation. The reverse, though, is not the case.
Composable	When trust is propagated, a particular agent may be connected to multiple trust chains. To come up with a final decision whether to trust or distrust this agent, the trust information received from the different chains need to be composed in order to build one aggregated picture. In this context, trust statements propagated from nodes close to oneself should have greater influence on the aggregated value than the ones from distant nodes (distance-based aging [45]). Composition is potentially difficult if the trust statements are contradictory [14].
Subjective	The subjective nature of trust becomes clear if one thinks about a review on Amazon [26]. A book review that totally reflects Alice's opinion will probably resolve in a high level of trust against the reviewer Rachel. Bob, however, who disagrees with the review, will have a lower trust in Rachel although it bases on the same evidence.
Fine-grained	Although trust is sometimes modeled in a binary manner (i.e. either trust or distrust), it is possible that Alice trusts both Bob and Claire but that she trusts Bob more than Claire. Hence, there may be multiple discrete levels of trust such as high, medium and low [41]. Mapped to numbers, trust may also be a continuous variable taking values within a certain interval (e.g. between 0 and 1).
Event-sensitive	It can take a long time to build trust. One negative experience, though, can destroy it [23].
Reflexive	Trust in oneself is always at the maximum value.
Self-reinforcing	It is human nature to preferentially interact with other agents that are trusted. Analogously, agents will avoid interacting with untrustworthy agents. Thus, the trustworthiness of other agents is inherently taken into consideration.

establishment [7]. Policy-based trust is often referred to as a *hard security mechanism* due to the exchange of hard evidence (e.g. credentials). Reputation-based trust, in contrast, is derived from the history of interactions. Hence, it can be seen as an estimation of trustworthiness (*soft security*). In this work, we focus on reputation-based trust. Reputation is defined as follows: "*Reputation is what is generally said or believed about a person's or thing's character or standing.*" [8].

It is based on referrals, ratings or reviews from members of a community. Therefore, it can be considered as a collective measure of trustworthiness [8]. Trustworthiness as a global value is objective. However, the trust an agent puts in someone or something as a combination of personal experience and referrals is subjective.

Research gap: design of reputation systems with reuse

It has been argued (e.g. by [3]) that most reputation-based trust models proposed in the academic community are built from scratch and do not rely on existing approaches. Only a few authors continue their research on the ideas of others. Thus, many approved models and promising thoughts go unregarded. The benefits of reuse, though, have been recognized in software engineering for years. However, there are only very few works that proposed single components to enhance existing approaches. Rehak et al. [9], for instance, introduced a generic mechanism that can be combined with existing trust models to extend their capabilities by efficiently modeling context. The benefits of such a component that can easily be combined with existing systems are obvious. Nonetheless, research in trust and reputation still lacks in sound and accepted principles to foster reuse.

To gradually close this gap, we aim to provide a framework for the design of new reputation systems with reuse. As described above, we thereto propose a hierarchical component taxonomy of computation engines used in reputation systems. Based on this taxonomy, we set up a repository containing design knowledge on both a conceptual and an implementation level. On the one hand, the uniform and well-structured artifacts collected in this repository can be used by developers to select, understand and apply existing concepts. On the other hand, they may encourage researchers to provide novel components on a conceptual and an implementation level. In this way, the reuse of ideas, concepts and implemented components as well as the communication of reuse knowledge should be achieved. Furthermore, we argue that the reusable components we identify in this work could extend current reputation models by the ability to gradually include the properties of trust described above. To evaluate whether our taxonomy/framework can cover all aspects of trust, we finally provide a table matching our component classes with trust properties.

A hierarchical component taxonomy for computation methods in reputation systems

To derive a taxonomy from existing models, our research includes two steps: (1) the analysis of the generic process of reputation systems and (2) the identification of logical components of the computation methods used in common trust and reputation models. A critical question is how to determine and classify single components. Thereto, we follow an approach to function-based component classification, which means that the taxonomy is derived from the functions the identified components fulfill.

The generic process of reputation systems

The generic process of reputation systems, as depicted in Figure 1, can be divided into three steps: (1) *collection* & *preparation*, (2) *computation* and (3) *storage* & *communication*. These steps are adapted from the three fundamental phases of reputation systems identified by [10] and [11]: feedback generation/collection, feedback aggregation and feedback distribution. Feedback aggregation as the central part of every trust and reputation system is furthermore divided into the three process steps *filtering*, *weighting* and *aggregation* taken together as computation. The context setting consists of a trustor who wants to build a trust relation toward a trustee by providing context and personalization parameters and receiving a trustee's reputation value.

Collection and preparation

In the collection and preparation phase, the reputation system gleans information about the past behavior of a trustee and prepares it for subsequent computing. Although personal experience is the most reliable, it is often not sufficiently available or nonexistent. Therefore, data from other sources needs to be collected. These can be various, ranging from public or personal collections of data centrally stored to data requested from different peers in a distributed network. After all available data is gathered, it is prepared for further use. Preparation techniques include normalization, for instance, which brings the input data from different sources into a uniform format. Once the preparation is completed, the reputation data serves as input for the computation phase.

Computation

The computation phase is the central part of every reputation system and takes the reputation information collected as input and generates a trust/reputation value as output. This phase can be divided into the three generic process steps *filtering*, *weighting* and *aggregation*. Depending on the computation engine, not all steps have to be implemented. The first two steps (filtering and weighting) preprocess the data for the subsequent aggregation. The need for these steps is obvious: The first question to be answered is *which* information is useful for further processing (filtering). The second process step concerns the question of *how relevant* the information is for the specific situation (weighting). In line with this, Zhang et al. [12] pointed out that current trust models can be classified into the two broad categories *filtering-based* and *discounting-based*. The difference between filtering and weighting is that the filtering process reduces the information amount while

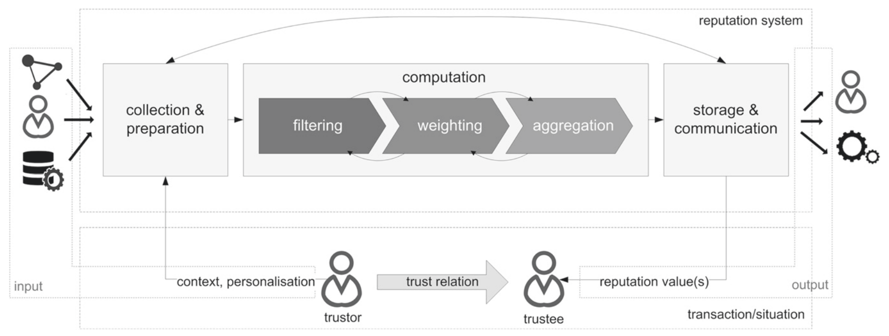

Figure 1 Generic process of a reputation system, inspired by [10].

it is enriched by weight factors in the second case. Therefore, filtering can be seen as *hard selection* while weighting is more like a *soft selection*. Finally, the reputation values are aggregated to calculate one or several reputation scores. Depending on the algorithm, the whole computation process or single process steps can be run through for multiple times.

Storage and communication

After reputation scores are calculated, they are either stored locally, in a public storage or both depending on the structure (centralized/decentralized/hybrid) of the reputation system. Common reputation systems not only provide the reputation scores but also offer extra information to help the end-users understand the meaning of a score. They should furthermore reveal the computation process to accomplish transparency.

In this work, we focus on the computation phase, since the first phase (collection & preparation) and the last phase (storage & communication) strongly depend on the structure of the reputation system (centralized or decentralized). The computation phase, however, is independent of the structure and can look alike for systems implemented in both centralized and decentralized environments. Therefore, it works well for design with reuse.

Hierarchical component taxonomy

In this section, the computation process is examined in detail. We introduce a novel hierarchical component taxonomy that is based on the functional blocks of common reputation systems identified in this work. Thereto, we clarify the objectives of the identified classes (functions) and name common examples. Our analysis and selection of reputation systems is based on different surveys [2,3,8,13,14]. Figure 2 gives an overview of the primary and secondary classes identified.

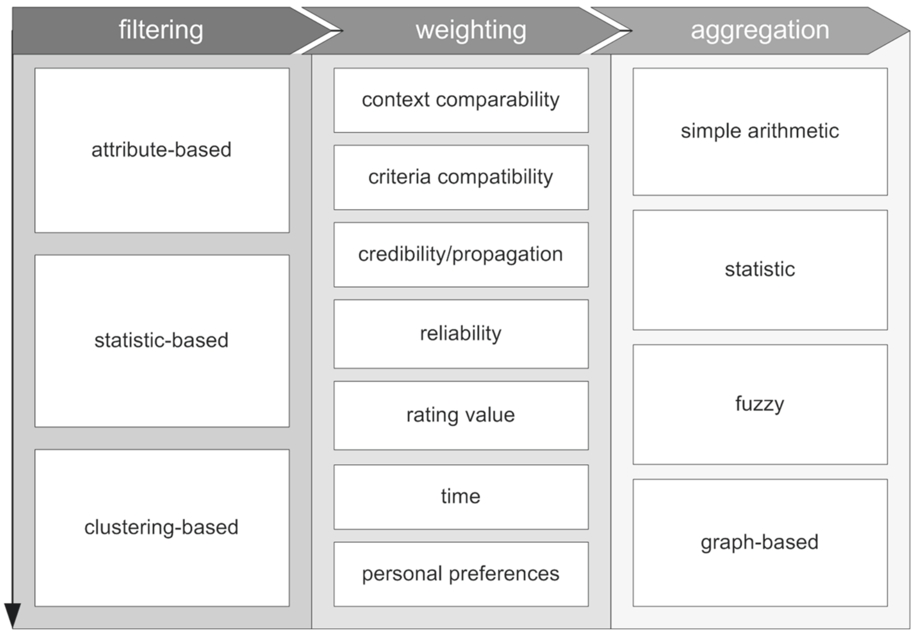

Figure 2 Classes of filtering-, weighting- and aggregation-techniques.

Beginning with the filtering phase, the three broad classes *attribute-based*, *statistic-based* and *clustering-based* filtering can be identified:

1. **Attribute-based filtering**: In several trust models, input data is filtered based on a constraint-factor defined for the value of single attributes. Attribute-based filters mostly implement a very simple logic, in which an attribute is usually compared to a reference value. Due to their lightweight, they are proper for reducing huge amounts of input data to the part necessary for the reputation calculation. Besides the initial filtering of input data, it is often applied after the weighting phase in order to filter referrals that have been strongly discounted. Time is an example of an attribute that is often constrained because it is desirable to disregard very old ratings. eBay's reputation system, for instance, only considers transactions having occurred in the last 12 months for their overview of positive, neutral and negative ratings. Other models such as Sporas [15] ignore every referral but the latest, if one party rated another party more than once. In this way, simple ballot stuffing attacks can be prevented. In ballot stuffing attacks, parties improve their reputation by means of positive ratings after fake transactions.
2. **Statistic-based filtering**: Further techniques that are used to enhance the robustness of trust models against the spread of false rumors apply statistical patterns. Whitby et al. [16], for example, proposed a statistical filter technique to filter out unfair ratings in Bayesian reputation systems applying the majority rule. The majority rule considers feedback that is far away from the majority's referrals as dishonest. In this way, dishonest or false feedback can easily be detected and filtered.
3. **Clustering-based filtering**: Clustering-based filter use cluster analysis approaches to identify unfair ratings. These approaches are comparatively expensive and therefore rarely used as filtering techniques. An exemplary procedure is to analyze an advisor's history. Since a rater never lies to himself, an obvious way to detect false ratings is to compare own experience with the advisor's referrals. Thus, both fair and unfair ratings can be identified. iCLUB [17], for example, calculates clusters of advisors whose evaluations against other parties are alike. Then, the cluster being most similar to the own opinion is chosen as fair ratings. If there is no common experience (e.g. bootstrapping), the majority rule will be applied. Another example for an approach using cluster filtering was proposed by Dellarocas [18].

Once all available information is reduced to those suitable for measuring trust and reputation in the current situation, it becomes clear that various data differ in their characteristics (e.g. context, reliability). Hence, the referrals are weighted in the second process step based on different factors. In contrast to the filtering step, applied techniques differ strongly. For that reason, our classification of weighting techniques is based on the properties of referrals that are analyzed for the discounting. We distinguish between the following classes:

1. **Context comparability**: Reputation data is always bound to the specific context in which it is created. Ratings that are generated in one application area might not be automatically applicable in another application area. In e-commerce, for instance, transactions are accomplished involving different prices, product types, payment

methods, quality or time. The non-consideration of this context leads to the value imbalance problem where a malicious seller can build a high reputation by selling cheap products while cheating on expensive ones. To increase comparability and avoid such situations, context has become a crucial attribute for many current approaches like [19] or [9].

2. **Criteria comparability**: Besides the context in which feedback is created, the criteria that underlie the evaluation are important. Particularly, if referrals from different application areas or communities are integrated, criteria comparability can be crucial. In file-sharing networks, for instance, a positive rating is often granted with a successful transaction independent of the quality of service. On e-commerce platforms, in contrast, quality may be a critical factor for customer satisfaction. Other distinctions could be the costs of reviews, the level of anonymity or the number of peers in different communities or application areas. Weighting based on criteria comparability can compensate these differences.
3. **Credibility/propagation**: In network structures such as in the web-of-trust, trust can be established along a recommendation or trust chain. Obviously, referrals that have first-hand information about the trustworthiness of an agent are more credible than referrals received at second-hand (with propagation degree of two) or higher. Therefore, several models apply a propagation (transitivity) rate to discount referrals based on their distance. The biometric identity trust model [20], for instance, derives the reputation-factor from the distance of nodes in a web-of-trust.
4. **Reliability**: Reliability or honesty of referrals can strongly affect the weight of reviews. The concept of feedback reputation that measures the agents' reliability in terms of providing honest feedback is often applied. As a consequence, referrals created by agents having a low feedback reputation have a low impact on the aggregated reputation. The bases for this calculation can be various. Google's PageRank [21], for instance, involves the position of every website connected to the trustee in the web graph in their recursive algorithm. Epinions [22], on the other hand, allows users to directly rate reviews and reviewers. In this way, the effects of unfair ratings are diminished.
5. **Rating value**: Trust is event sensitive. For stronger punishment of bad behavior, the weight of positive ratings compared to negative ratings can be calculated asymmetrically. An example for a model using an "adaptive forgetting scheme" was proposed by Sun et al. [23], in which good reputation can be built slowly through good behavior but easily be ruined through bad behavior.
6. **Time**: Due to the dynamic nature of trust, it has been widely recognized that time is one important factor for the weighting of referrals. Old feedback might not be as relevant for reputation scoring as new referrals. An example measure for time-based weighting is the "forgetting factor" proposed by Jøsang [24].
7. **Personal preferences**: Reputation systems are used by various end-users (e.g. human decision makers, services). Therefore, a reputation system must allow the adaptation of its techniques to subjective personal preferences. Different actors might have different perceptions regarding the importance of direct experience and referrals, the significance of distinct information sources or the rating of newcomers.

The tuple of reputation data and weight-factor(s) serve as input for the third step of the computation process - the aggregation. In this phase, one or several trust/reputation values are calculated by composing the available information. In some cases, the weighting and the aggregation process are run through repetitively in an iterative manner. However, the single steps can still be logically separated. The list of proposed algorithms to aggregate trust and reputation values has become very long during the last decade. Here, we summarize the most common aggregation techniques and classify them into the four blocks *simple arithmetic*, *statistic*, *fuzzy* and *graph-based models*:

1. **Simple arithmetic**: The first class includes simple aggregation techniques like ranking, summation or average. Ranking is a very basic way to measure trustworthiness. In ranking algorithms, ratings are counted and organized in a descending order based on that value. This measure has no exact reputation score. Instead, it is frequently used as a proxy for the relative importance/trustworthiness. Examples for systems using ranking algorithms are message boards like Slashdot [25] or citation counts used to calculate the impact factor in academic literature. Other aggregation techniques that are well known due to the implementation on eBay or Amazon [26] are the summation (adding up positive and negative ratings) or the average of ratings. Summation, though, can easily be misleading, since a value of 90 does not reveal the composition of positive and negative ratings (e.g. +100,-10 or +90,0). The average, on the other hand, is a very intuitive and easily understandable algorithm.
2. **Statistic**: Many of the prominent trust models proposed in the last years use a statistical approach to provide a solid mathematical basis for trust management. Applied techniques range from *Bayesian probability* over *belief models* to *Hidden Markov Models*. All models based on the beta probability density function (beta PDF) are examples for models simply using Bayesian probability. The beta PDF represents the probability distributions of binary events. The *a priori* reputation score is thereby gradually updated by new ratings. The result is a reputation score that is described in a beta PDF function parameter tuple (α, β), whereby α represents positive and β represents negative ratings. A well known model using the beta PDF is the Beta Reputation system [24]. A weakness of Bayesian probabilistic models, however, is that they cannot handle uncertainty. Therefore, belief models extend the probabilistic approach by Dempster-Shafer theory (DST) or subjective logic to include the notion of uncertainty. Trust and reputation models involving a belief model were proposed by Jøsang [27] or Yu and Singh [28]. More complex solutions that are based on machine learning, use the Hidden Markov Model, a generalization of the beta model, to better cope with the dynamic behavior. An example was introduced by Malik et al. [29].
3. **Fuzzy**: Aggregation techniques classified as fuzzy models use fuzzy logic to calculate a reputation value. In contrast to classical logic, fuzzy logic allows to model truth or falsity within an interval of [0,1]. Thus, it can describe the degree to which an agent/resource is trustworthy or not trustworthy. Fuzzy logic has been proven to deal well with uncertainty and mimic the human decision making process [30]. Thereby, a linguistic approach is often applied. REGRET [31] is one prominent example of a trust model making use of fuzzy logic.

4. **Graph-based**: A variety of trust models employ a graph-based approach. They rely on different measures describing the position of nodes in a network involving the flow of transitive trust along trust chains in network structures. As online social networks have become popular as a medium for disseminating information and connecting people, many models regarding trust in social networks have lately been proposed. Graph-based approaches use measures from the field of graph theory such as centrality (e.g. Eigenvector, betweenness), distance or node-degree. Reputation values, for instance, grow with the number of incoming edges (in-degree) and increase or decrease with the number of outgoing edges (out-degree). The impact of one edge on the overall reputation can depend on several factors like the reputation of the node an edge comes from or the distance of two nodes. Popular algorithms using graph-based flow model are Google's PageRank [21] as well as the Eigentrust Algorithm [32]. Other examples are the web-of-trust or trust models particularly designed for social networks as described in [14]. As mentioned above, the weighting and aggregation phases are incrementally run through for several times due to the incremental nature of these algorithms.

The classification of the computation engine's components used in different trust models in this taxonomy is not limited to one component of each primary class. Depending on the computation process, several filtering, weighting and aggregation techniques can be combined and run through more than once. Malik et al. [29], for instance, introduced a hybrid model combining heuristic and statistical approaches. However, our taxonomy can reveal the single logical components a computation engine is built on. Moreover, it serves as an overview of existing approaches. Since every currently known reputation system can find its position, to the best of our knowledge, this taxonomy can be seen as complete. Though, an extension by new classes driven by novel models and ideas is possible. Our hierarchical component taxonomy currently contains 3 primary component classes, 14 secondary component classes, 23 component terms and 29 subsets. Table 2 shows an excerpt of the hierarchical component taxonomy with building blocks of the primary class "weighting". The full taxonomy is provided in Additional file 1: Table S1.

The component taxonomy as a framework for design with reuse

The hierarchical component taxonomy introduced in the former section serves as a natural framework for the design of reputation systems with reuse. To support this process, we set up a component repository combining a knowledge and a service repository. Thus, it does not only contain information about software components on implementation level but also provides extensive descriptions of the ideas applied on a conceptual level. This comprehensive set of fundamental component concepts and ideas combined with the related implementation allows the reuse of both ideas and already implemented components.

In this section, we firstly describe the conceptual design of our component repository in detail. Then, we elaborate on the implementation of a web application employing our thorough repository to provide design knowledge for reuse on a conceptual and an implementation level.

Table 2 Excerpt of the hierarchical component taxonomy with descriptions

Primary component class	Secondary component class	Component term	Subset	Description
weighting	credibility/ propagation	propagation discount		Discount referrals along trust chains
	reliability	subjective reliability	property similarity	Discount based on similarity of personal properties
			rating similarity	Discount referrals based on similarity of ratings toward other agents
		objective reliability	Explicit	Discount based on explicit reputation information like referrals or certificates
			Implicit	Discount based on implicit reputation information like profile age, number of referrals or position
	rating value	asymmetric rating		Strongly discount positive ratings compared to negative ratings (event sensitive)
...	...	...	...	...

Conceptual design of the component repository

Reuse-based software engineering can be implemented on different levels of abstraction, ranging from the reuse of ideas to the reuse of already implemented software components for a very specific application area. In this work, we want to apply our taxonomy for reuse on two levels – a conceptual level and an implementation level. Therefore, the developed repository provides design knowledge for reuse on two logical layers (see Figure 3).

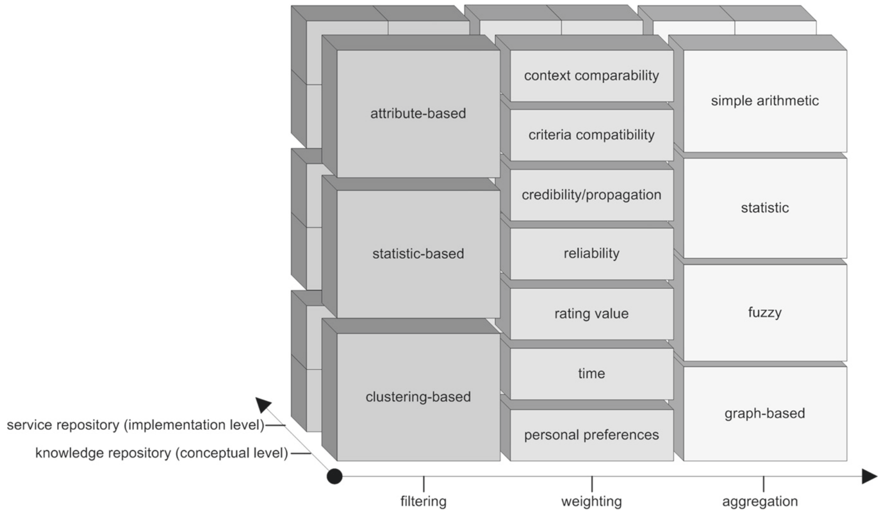

Figure 3 Logical layers of the component repository for design with reuse.

Reuse on conceptual level

When reusing an implemented component, one is unavoidably constrained by design decisions that have been made by the developer. A way to prevent this is to conceive more abstract designs that do not specify the implementation. Thus, we provide an abstract solution to a problem by means of design pattern-like concepts. Design patterns are descriptions of commonly occurring problems and a generic solution to the problems that can be used in different settings [33]. Our design pattern-like concepts consist of essential elements that are exemplary depicted in Table 3.

Reuse on implementation level

On implementation level, we provide fully implemented reusable components by means of web services in a service-orientated architecture. These services encapsulate the concepts' logic and functionality in independent and interchangeable modules to achieve the separation of concerns. The web services are incorporated via well-defined interfaces. All services provided are registered as artifacts in the service repository. An artifact contains essential information about one live reachable service such as ID, type (REST or ws), URL, description, parameters, example calls, example output, the design pattern that is implemented by the service, and tags describing the functionality. Table 4 shows an example artifact for the design pattern described above.

Table 3 Design pattern on the conceptual level (example)

Component term	**Context similarity**
Subset	**Absolute congruence**
Description	This component uses an absolute congruence metric as similarity measure to identify context similarity.
Problem description	Reputation data is always bound to the specific context in which it was created. Ratings that were generated in one application area might not be automatically applicable in another application area which can result in the value imbalance problem.
Solution description	Apply similarity measurement between context c_i (reference context) and context c_j of referrals in the referral set to deliver a weight-factor for each item of the referral set using the following formula: $$w(c_1, c_2) := \frac{k(c_i) \cap k(c_j)}{k(c_i) \cup k(c_j)}$$ $k(c_i)$ denotes the total number of keywords describing context c_i.
Applicability	Set of nominal context attributes.
Code example (php)	`function calculate_values($reference, $context_sets) {` `$reference_context = $reference['context_attributes'];` `$return_values = array();` `while(!empty($context_sets)) {` `...shortened...` `}` `return $return_values;` `}`
Implementation	Context similarity-based weighting service (absolute congruence)
Literature	• Mohammad Gias Uddin, Mohammad Zulkernine, and Sheikh Iqbal Ahamed. 2008. CAT: a context-aware trust model for open and dynamic systems. In Proceedings of the 2008 ACM symposium on Applied computing (SAC '08). ACM, New York, NY, USA, 2024–2029.
Tags	weighting, context, similarity, congruence

Table 4 Web service description on implementation level (example)

Component term	**Context similarity**
Subset	**Absolute congruence**
Type	REST
Demo	http://trust.bayforsec.de/ngot/webservice/Client/?=Weighting-congruence-absolute-call.php
Description	This service provides an absolute similarity measurement between a reference context and a context-set of referrals. Example: The sets 'registered','charged', 'verified' and 'registered', 'costless', 'unverified' have a similarity of 1/3.

Parameters

```
// define words that describe the quality of a referall
$reference_context : array("words" => array (TEXT));

$referral_sets = array( $context_set );

$referral_set = array( "id" => NUMBER, "words" => array (TEXT));
```

Example call

```
require_once('WebserviceCallHelper.php');

$arguments = array("words" => array ("registered","charged","verified"));
$referral_set = array();
$referral_set[0] = array( "id" => "10000", "words" =>
      array ("registered","costless", "unverified"));
$referral_set[1] = array( "id" => "10001", "words" =>
      array ("registered","charged", "verified"));
$referral_set[2] = array( "id" => "10002", "words" =>
      array ("registered","costless", "verified"));

$webservice_call = new WebserviceCallHelper(array(
        'base_url' => WEBSERVICE_URL,
        'format' => "html",
        'component' => "Weighting\CongruenceAbsolute"
));
$webservice_call->get_result($arguments, $referral_set);
```

Example output

```
Array
(
    [status] => 200
    [data] => Array
        (
            [0] => Array
                (
                    [0] => 10000
                    [1] => 0.2
                )
            [1] => Array
                (
                    [0] => 10001
                    [1] => 1
                )
            [2] => Array
                (
                    [0] => 10002
                    [1] => 0.5
                )
        )
)
```

Pattern Implemented	Context similarity (Absolute congruence)
Tags	weighting, context, similarity, congruence

Implementation of the repository

To demonstrate the feasibility of our approach, we have prototypically implemented the repository as a web-based application in a three-tier client-server-architecture [34]. To give an overview of the chosen architecture, we distinguish between server-side and client-side implementation.

Server-side

On server-side, the logic is implemented in PHP on an Apache server (logic layer) connecting to a MySQL database (persistent layer). The MySQL database contains all data regarding the design patterns as described in Table 2. Each of these design patterns is also implemented in a web service. To enable a standardized realization of new web services and a flawless call via standardized interfaces, we employ an abstract class *Component*. All components (implemented as web services) must inherit from this class, which particularly requires overwriting the function *calculate_values*. To make the generic component independent of the input data, developers are advised to make use of the PHP function *func_get_args()*. In this way, distinct components can receive a variety of arguments. To consistently handle client calls, our architecture is extended by a *WebserviceCallHandler*. Figure 4 depicts the schematic layout.

All web services implementing the trust pattern currently described in our knowledge repository have been created and registered as artifacts in our service repository [34]. Furthermore, these artifacts are described in detail including a definition of input, output and example calls as defined in Table 4.

Client-side

On client-side (presentation layer), we employ the current web standards HTML5, JavaScript and CSS (Bootstrap). The front end is divided into three main pages – "overview", "knowledge repository" and "service repository" – which provide information on the general concept, the trust patterns and the web services. To enable a standardized call of a web service from client-side, a *WebserviceCallHelper* allows a simple call of each component by configuration and provides all functions necessary to establish a connection to the repository. The configuration details are passed to the constructor, which requires a *base_url*, an output *format* (HTML, XML or JSON) and a unique *component* name as illustrated below.

Example call of a filtering component via the WebserviceCallHelper

```
$webservice_call = new WebserviceCallHelper(array(
            'base_url' => "http://trust.bayforsec.de/ngot/webservice/",
            'format' => "html",
            'component' => "Filtering\AgeBasedAbsolute"
        ));
$webservice_call->get_result($arguments, $referral_set);
```

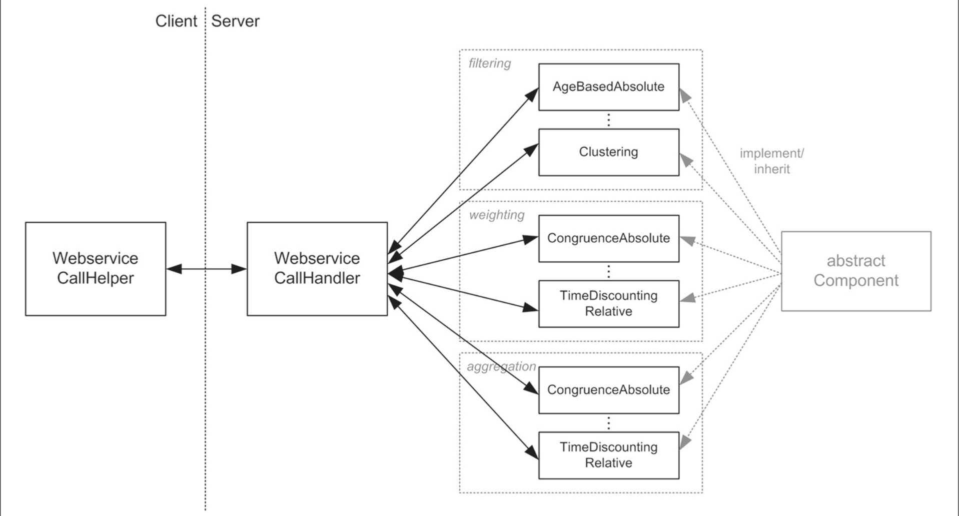

Figure 4 Schematic view on the service architecture.

Evaluation

To rigorously demonstrate the proper functioning and quality of our approach, we carry out a two-part evaluation of our artifact in this section. As there is currently no comparable framework, we firstly perform a descriptive scenario-based evaluation. According to Hevner et al. [4], this is a standard approach for innovative artifacts like ours. To demonstrate the completeness of our taxonomy, we secondly conduct a static analysis [4], in which we match all components to the trust properties described in Section 'The notion of trust and its properties'.

Scenario analysis: Reputation system development

The fictitious web developer John Gray runs an electronic marketplace platform for philatelists and numismatics. The platform has been launched with his friends as the first users but has been growing fast. Meanwhile, most users do not know each other in person anymore. As a result, many of the initial users have stopped interacting with the newcomers as they do not trust them. After realizing this problem, John decides to introduce an online reputation system in order to establish trust among the strangers. In the following, we describe how our knowledge and service repository can help him to build a reputation system that perfectly meets his requirements.

Having read the basics on our component model, John concludes that he wants to build a computation engine that makes use of components of all three phases – filtering, weighting and aggregation. Thinking about the experiences made with sellers on the platform, he recognizes that most of them do not deliver the same quality all the time. Thus, an *age-based filter* should be employed to make old referrals less important than new ones. Furthermore, there are sellers that usually deliver high quality stamps while offering poor quality coins. Therefore, a weighting component based on *context similarity (absolute congruence)* should be selected. Regarding the aggregation alternatives, John decides to make use of the *average* component as the simple average is probably the most intuitive and most transparent aggregation technique for the users. Finally, the single components are combined in sequence to a fully functional computation engine as depicted in Figure 5.

The code listed below shows an example for the implementation of John's computation engine. Here, the WebserviceCallHelpers for each of the selected components have to be instantiated first as described in Section 'Client-side'. Secondly, the referral set needs to be loaded and prepared according to input parameter descriptions provided for each web service in the component repository. In its current form, the framework does not provide any classes to automatically plug single components together (glue class). Thus, the developer has to ensure that the output of one component is correctly provided as an input for the following component. This lose coupling, however, allows for more flexibility. All details on the input and output format can be found in the artifact description of each component. While the input data varies from component to component, the output is alike for components of each primary class.

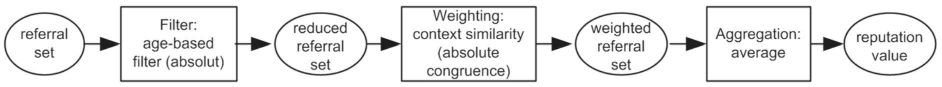

Figure 5 Sequence of components for John's computation engine.

Example code for the computation engine described above

```
//create helper for filtering component
$webservice_filter = new WebserviceCallHelper(array(
            'base_url' => WEBSERVICE_URL,
            'format' => "html",
            'component' => "Filtering\AgeBasedAbsolute"
        ));
//create helper for weighting component
$webservice_weight = ...as above...
//create helper for aggregation component
$webservice_aggregate = ...as above...
//define referral_set
$referral_set = array("id" => NUMBER, "time" => DATE, "context" => array (TEXT), "
     rating" => NUMBER);
//call computation
$reputation_value = $webservice_aggregate(
                $glue->prepare_for_aggregation($webservice_weight(
                    $glue->prepare_for_weighting($webservice_filter->get_result(
                        $referral_set))
                            ))
                    );
```

This scenario elucidates that our knowledge and service repository has an obvious utility from a practical point of view since developers can easily access it and gain knowledge about online reputation systems. Thereby, we may help to better spread innovative ideas and allow developers to experiment with different computation techniques. However, our approach requires specific knowledge on the structure of the repository, the functioning of each component and details on how to plug components together. Developers need to manually combine components and take care, whether they use valid input data and a feasible combination of reputation system components. In its current form, our framework is not very "developer friendly". Therefore, further research will be necessary in order to improve the practical usability of our component repository. In Section 'Contribution and future work', we discuss open issues in more detail.

Static analysis: Matching components and trust properties

To guarantee the proper generation of computational trust, trust-enforcing mechanisms such as reputation systems should be able to consider and address all properties of trust. As our component taxonomy serves as a framework for the design of new reputation systems with reuse, it should enable developers to extend current reputation models by the ability to gradually include the various properties of trust. Therefore, a way to evaluate the completeness of our solution is to review whether a trust system that is built according to our framework could meet this standard. In Table 5, we match the computation components identified above to the properties of trust introduced in Section 'The notion of trust and its properties'. Since our taxonomy is based on reputation systems analyzed in various surveys, this approach also enables us to identify aspects of trust that have only rarely been considered in research so far.

Examining Table 5, we find that all trust properties listed are widely covered. There is at least one component addressing each single characteristic. Going into more detail, we find that there are many proposals that have developed components to personalize reputation systems, thus covering the subjective property of trust. This reflects a general trend to an enhanced personalization of reputation systems. Furthermore, it becomes clear that all of our weighting and aggregation components follow the fine-grained property of trust, i.e. that trust can be modeled as a continuous variable. For the filtering components, the fine-grained property is not entirely applicable in the same meaning as filtering has no direct influence on the trust value of referrals. Since the effect of filtering is that a referral

Table 5 Matching reputation system components and trust properties

			Trust properties									
	Primary component class	**Secondary component class**	**Dynamic**	**Context-dependent**	**Multi-faceted**	**Propagative**	**Composable**	**Subjective**	**Fine-grained**	**Event-sensitive**	**Reflexive**	**Self-reinforcing**
Computation components	**Filtering**	**attribute-based**	●	○	○	○	◐	●	○	○	○	○
		statistic-based	●	○	○	○	○	●	○	○	○	◐
		clustering	○	○	○	○	○	●	○	○	●	◐
	Weighting	**context comparability**	◐	●	◐	○	○	○	●	○	○	○
		criteria comparability	○	◐	●	○	○	●	●	○	○	○
		credibility	○	○	○	●	●	●	●	○	◐	○
		reliability	○	○	○	◐	◐	●	●	○	◐	●
		rating value	○	○	○	○	○	○	●	●	○	◐
		time	●	○	○	○	○	○	●	○	○	○
		personal preferences	○	○	○	○	○	●	●	○	◐	○
	Aggregation	**simple arithmetic**	○	○	○	○	●	○	●	○	○	◐
		statistic	○	○	○	○	●	○	●	○	○	◐
		fuzzy	○	○	○	○	●	○	●	○	○	◐
		graph-based	○	○	○	○	●	◐	●	○	○	◐

○ not addressed.
◐ partly addressed.
● completely addressed.

either is further considered or not, it shows a binary character rather than being fine-grained. In contrast to the subjective and fine-grained properties of trust, other properties such as context-dependent, multi-faceted and event-sensitive are particularly addressed by only one or two components. Note that this does not automatically mean that there is an increased necessity for future research concerning these properties. It may also be possible that one component is enough to cover one trust property. More detailed studies on this could be part of future work.

Overall, we can say that computational trust can be represented quite accurately when using our taxonomy and the provided components as a basis for the development of new reputation systems or the extension of existing models. Note, however, that this is only one view on our taxonomy. Conducting a comparable analysis from the viewpoint of attacks and defense mechanisms, for instance, the outcomes may vary greatly.

Contribution and future work

Many surveys of trust and reputation systems give an overview of existing trust and reputation systems by means of a classification of existing models and approaches. In contrast to this, we provide a collection of ideas and concepts classified by their functions. Furthermore, these ideas are not only named but also clearly described in well-structured design pattern-like artifacts which can easily be adapted to a specific situation. Therewith, we reorganized the design knowledge for computation techniques in reputation systems and translated the most common ideas into a uniform format. To directly make use of novel components, the web services created on implementation level can instantly be reused and integrated in existing reputation systems to extend their capabilities. This approach (i.e. publicly providing implemented computation components as web services) may help to better spread innovative ideas in trust and reputation systems and give system builders a better choice allowing to experiment with different computation techniques. Moreover, we encourage researchers to focus on the design of single components by providing a platform on which concepts and their prototypical implementation can be made publicly available.

Nonetheless, there are still some unexplored areas regarding the design with reuse in trust and reputation systems. Firstly, reusability could play a role in process steps other than the computation phase. To clarify the opportunities, further research is necessary in this area. Secondly, our hierarchical taxonomy is currently limited to a functional view on the identified components but developers may also benefit from additional views. Because of the importance of the robustness of trust and reputation systems [35], we are particularly interested in an attack view. In [36], we present first ideas on this issue. We propose a taxonomy of attacks on reputation systems and then refer to the single components of our repository as solutions to the specific attack classes. In this way, we not only support reputation system designers in the development of more reliable and more robust reputation systems with already existing components but also help to identify weaknesses that have not been addressed so far. Thirdly, the selection and interpretation of adequate components for new reputation systems in a particular application area requires time, effort and – to some extent – knowledge of this research area. To increase usability, a software application is needed to support a user in this development process. Ultimately, the application may even be able to automatically find the most qualified composition for specific requirements and input data. This, in turn, demands for generic testbeds that enable objective evaluations of reputation systems because so far, researchers have mainly been developing their own testing scenarios favoring their own work [37]. The most well-known proposals regarding independent testbeds are ART [38] and TREET [37]. Recently, Irissappane and Zhang [39,40] made another important step forward by introducing a publicly available testbed that is able to reflect real environmental settings. We plan to use their tool in future studies. Finally, we need to observe the usage of our repository in practice to learn from how users deal with it. This can either be done through conducting experimental user studies or by interviewing developers who use our repository in a real environment. In this way, we can run through a continuous improvement process.

Conclusion

The research in trust and reputation systems is still growing. In this paper, we presented concepts to foster reuse of existing approaches. We provided a hierarchical taxonomy of computation components from a functional view and described the implementation of a component repository that serves as both a knowledge base and a service repository. In this way, we communicate design knowledge for reuse, support the development of new reputation systems and encourage researchers to focus on the development of single components that can be integrated in various reputation systems to easily extend their capabilities by new features. Matching the identified components and the properties of trust, we found that integrating existing ideas and concepts can lead to a reputation system that widely reflects computational trust by addressing all properties of trust described in literature.

Additional file

Additional file 1: Table S1.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
JS proposed the initial idea of this paper. He developed the hierarchical component taxonomy and implemented the component repository. CR and JS conducted the evaluation of the proposed ideas. CR and JS furthermore revised this paper according to reviewers' comments. GP supervised the research, contributed to the paper writing and made suggestions. All authors read and approved the final manuscript.

Acknowledgment
The research leading to these results was supported by the Bavarian State Ministry of Education, Science and the Arts" as part of the FORSEC research association.

Received: 9 December 2014 Accepted: 1 April 2015
Published online: 13 May 2015

References
1. Electronics, Cars, Fashion, Collectibles, Coupons and More | eBay. http://www.ebay.com
2. Yao Y, Ruohomaa S, Xu F (2012) Addressing common vulnerabilities of reputation systems for electronic commerce. J Theor Appl Electron Commerce Res 7(1):1–20
3. Tavakolifard M, Almeroth KC (2012) A taxonomy to express open challenges in trust and reputation systems. J Commun 7(7):538–551
4. Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Quarterly 28(1):75–105
5. McKnight DH, Chervany NL (1996) The Meanings of Trust. Technical report. University of Minnesota, Management Information Systems Research Center
6. Gambetta D (1988) Can we trust trust? In: Gambetta D (ed). Trust: making and breaking cooperative relations. Basil Blackwell, Oxford. pp 213–237
7. Artz D, Gil Y (2007) A survey of trust in computer science and the semantic web. Web Semantics 5(2):58–71
8. Jøsang A, Ismail R, Boyd C (2007) A survey of trust and reputation systems for online service provision. Decis Support Syst 43(2):618–644

9. Rehak M, Gregor M, Pechoucek M, Bradshaw J (2006) Representing context for multiagent trust modeling. In: Skowron A, Barthès JP, Jain LC, Sun R, Morizet-Mahoudeaux P, Liu J, Zhong N (eds). Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Hong Kong, China. IEEE Computer Society, Washington, DC. pp 737–746
10. Swamynathan G, Almeroth KC, Zhao BY (2010) The design of a reliable reputation system. Electron Commerce Res 10(3–4):239–270
11. Resnick P, Kuwabara K, Zeckhauser R, Friedman E (2000) Reputation systems. Commun ACM 43(12):45–48
12. Zhang L, Jiang S, Zhang J, Ng WK (2012) Robustness of trust models and combinations for handling unfair ratings. In: Dimitrakos T, Moona R, Patel D, McKnight DH (eds). Trust Management VI: Proceedings of the 6th IFIP WG 11.11 international conference (IFIPTM). Springer, Berlin, Heidelberg, Surat, India. pp 36–51
13. Noorian Z, Ulieru M (2010) The state of the art in trust and reputation systems: a framework for comparison. J Theor Appl Electron Commerce Res 5(2):97–117
14. Sherchan W, Nepal S, Paris C (2013) A survey of trust in social networks. ACM Comput Surv 45(4):1–33
15. Zacharia G, Moukas A, Maes P (2000) Collaborative reputation mechanisms for electronic marketplaces. Decis Support Syst 29(4):371–388
16. Whitby A, Jøsang A, Indulska J (2004) Filtering out unfair ratings in Bayesian reputation systems. In: Falcone R, Barber S, Sabater J, Singh M (eds). Proceedings of the third international joint conference on autonomous agents and multi agent systems, New, York, USA. IEEE Computer Society, Washington, DC. pp 106–117
17. Liu S, Zhang J, Miao C, Theng Y-L, Kot AC (2011) iCLUB: an integrated clustering-based approach to improve the robustness of reputation systems. In: Sonenberg L, Stone P, Tumer K, Yolum P (eds). Proceedings of the 10th international conference on Autonomous Agents and Multiagent Systems (AAMAS), Taipei, Taiwan. IFAAMAS, Richland, SC. pp 1151–1152
18. Dellarocas C (2000) Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: Jhingran A, MacKie J, Tygar D (eds). Proceedings of the 2nd ACM conference on electronic commerce, Minneapolis, MN. ACM, New York. pp 150–157
19. Zhang H, Wang Y, Zhang X (2012) A trust vector approach to transaction context-aware trust evaluation in e-commerce and e-service environments. In: Shih C, Son S, Kuo T, Huemer C (eds). Proceedings of the 5th IEEE international conference on Service-Oriented Computing and Applications (SOCA). IEEE Computer Society Washington, DC, Taipei, Taiwan. pp 1–8
20. Obergrusberger F, Baloglu B, Sänger J, Senk C (2013) Biometric identity trust: toward secure biometric enrollment in web environments. In: Yousif M, Schubert L (eds). Proceedings of the 3rd international conference on Cloud Computing (CloudComp), Vienna, Austria. Springer, Berlin, Heidelberg. pp 124–133
21. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Networks 30(1-7):107–177
22. Epinions.com: Read expert reviews on Electronics, Cars, Books, Movies, Music and More. http://www.epinions.com/
23. Sun Y, Han Z, Yu W, Ray Liu K (2006) Attacks on trust evaluation in distributed networks. In: Proceedings of Th 40th annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, IEEE Computer Society Washington, DC. pp 1461–1466
24. Jøsang A, Ismail R (2002) The beta reputation system. In: Proceedings of the 15th bled conference on electronic commerce, Bled, Slovenia. pp 41–55
25. Slashdot: News for nerds, stuff that matters. http://www.slashdot.org/
26. Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more. http://www.amazon.com
27. Jøsang A (2001) A logic for uncertain probabilities. Int J Uncertainty Fuzziness Knowledge-Based Syst 9(3):279–311
28. Yu B, Singh MP (2002) An evidential model of distributed reputation management. In: Proceedings of the first International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Bologna, Italy. ACM, New York, NY. pp 294–301
29. Malik Z, Akbar I, Bouguettaya A (2009) Web services reputation assessment using a Hidden Markov Model. In: Baresi L, Chi CH, Suzuki J (eds). Service-oriented computing: Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave), Stockholm, Sweden. Springer Berlin, Heidelberg. pp 576–591
30. Song S, Hwang K, Zhou R, Yu-Kwong K (2005) Trusted P2P transactions with fuzzy reputation aggregation, Vol. 9
31. Sabater J, Sierra C (2002) Reputation and social network analysis in multi-agent systems. In: Proceedings of the first International joint conference on Autonomous Agents and Multiagent Systems (AAMAS), Bologna, Italy. ACM, New York, NY. pp 475–482
32. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The Eigentrust algorithm for reputation management in P2P networks. In: Hencsey G, White B, Chen YF, Kovács L, Lawrence S (eds). Proceedings of the 12th International Conference on World Wide Web (WWW), Budapest, Hungary. ACM, New York, NY. pp 640–651
33. Gamma E (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading
34. Next Generation Online Trust. http://trust.bayforsec.de
35. Jøsang A (2012) Robustness of trust and reputation systems: does it matter? In: Dimitrakos T, Moona R, Patel D, McKnight DH (eds). Trust management VI: Proceedings of the 6th IFIP WG 11.11 International Conference (IFIPTM), Surat, India. Springer, Berlin, Heidelberg. pp 253–262
36. Sänger J, Pernul G (2015) Reusable defense components for online reputation systems. In: Marsh S, Jensen CD, Murayma Y, Dimitrakos T (eds). Trust management IX: Proceedings of the 9th IFIP WG 11.11 International Conference (IFIPTM), Hamburg, Germany. Springer, Berlin, Heidelberg
37. Kerr R, Cohen R (2010) TREET: The Trust and Reputation Experimentation and Evaluation Testbed. Electron Commerce Res 10(3–4):271–290
38. Fullam KK, Voss M, Klos TB, Muller G, Sabater J, Schlosser A, Topol Z, Barber KS, Rosenschein JS, Vercouter L (2005) A specification of the Agent Reputation and Trust (ART) Testbed. In: Dignum F, Dignum V, Koenig S, Kraus S, Singh MP, Wooldridge M (eds). Proceedings of the 4th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS), Utrecht, Netherlands. ACM, New York, NY, USA. pp 512–518

39. Irissappane AA, Jiang S, Zhang J (2012) Towards a comprehensive Testbed to evaluate the robustness of reputation systems against unfair rating attacks. In: Herder E, Yacef K, Chen L, Weibelzahl S (eds). Workshop and Poster Proceedings of the 20th conference on User Modeling, Adaptation, and Personalization (UMAP), Montreal, Canada. Springer Berlin, Heidelberg
40. Irissappane AA, Zhang J (2014) A Testbed to evaluate the robustness of reputation systems in e-Marketplaces. In: Bazzan A, Huhns MN, Lomuscio A, Scerri P (eds). Proceedings of the 13th international conference on Autonomous Agents and Multiagent Systems (AAMAS), Paris, France. ACM, New York, NY, USA. pp 1629–1630
41. Grandison T, Sloman M (2000) A survey of trust in internet applications. IEEE Commun Surv Tutorials 3(4):2–16
42. Yu B, Singh MP (2000) A social mechanism of reputation management in electronic communities. In: Goos G, Hartmanis J, van Leeuwen J (eds). Proceedings of the 4th international workshop on cooperative information agents IV - The future of information agents in cyberspace (CIA), Boston, USA, Springer, London, UK. pp 154–165
43. Wang Y, Vassileva J (2003) Trust and reputation model in peer-to-peer networks. In: Shahmehri N, Graham RL, Caronni G (eds). Proceedings of the 3rd international conference on Peer-to-Peer Computing (P2P), Linköping, Sweden. IEEE Computer Society Washington, DC. pp 150–157
44. Golbeck JA (2005) Computing and applying trust in web-based social networks. PhD thesis, University of Maryland, College Park, MD, USA
45. Haque MM, Ahamed SI (2007) An omnipresent Formal Trust Model (FTM) for pervasive computing environment. In: Proceedings of the 31st annual international Computer Software and Applications Conference (COMPSAC), Bejing, China. IEEE Computer Society Washington, DC. pp 49–56
46. Uddin MG, Zulkernine M, Ahamed SI (2008) CAT: a context-aware trust model for open and dynamic systems. In: Wainwright RL, Haddad H (eds). Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Brazil. ACM, New York, NY. pp 2024–2029
47. Wishart R, Robinson R, Indulska J, Jøsang A (2005) Superstringrep: reputation-enhanced service discovery. In: Estivill-Castro V (ed). Proceedings of the 28th Australasian conference on computer science, Newcastle, NSW, Australia. Australian Computer Society, Inc, Darlinghurst, Australia. pp 49–57
48. Christianson B, Harbison WS (1997) Why Isn't trust transitive? In: Christianson B, Crispo B, Lomas T, Roe M (eds). Proceedings of the 2nd international workshop on security protocols, Paris, France. Springer, London, UK. pp 171–176