SOCA (2009) 3:47–63
DOI 10.1007/s11761-009-0037-8

SPECIAL ISSUE PAPER

Collaboration through computation: incorporating trust model into service-based software systems

Mohammad Gias Uddin · Mohammad Zulkernine · Sheikh Iqbal Ahamed

Received: 27 May 2008 / Revised: 7 January 2009 / Accepted: 22 January 2009 / Published online: 20 February 2009

Abstract The open and dynamic nature of service-based software systems necessitates spontaneous and trustworthy interactions between collaborating entities. Service providers are exposed to users spanned across multiple organizational domains, so can be exploited by potentially untrustworthy service requestors. Given that, service providers need to trust requestors before granting them with services. Trust encompasses a number of quality attributes (e.g., security, competence, honesty) and helps in dynamic decision making. In this paper, we present a trust-based service collaboration approach, facilitated by the analysis of service-based interactions between service providers and requestors, and recommendations between service providers. Service providers exchange recommendations to convey their trust on requestors. This collaboration is quantified using our proposed trust model, called CAT, a Context-Aware Trust model based on service-based interactions by considering services as contexts. We identify a number of collaboration-based trust properties including risk and context-awareness and incorporate them in CAT. A context-similarity parameter is introduced to decide on similar services. A time-based ageing parameter is proposed to decrease trust values over time without any further interactions. Direct and indirect recommendations from other service providers are included in total trust calculation, with a path-based ageing parameter applying over indirect recommendations. A mechanism to calculate the accuracy of recommendations is proposed to differentiate between reliable and unreliable recommendations. These calculation schemes are employed in a trust-based service collaboration algorithm to automatically decide on granting services to requestors. The approach is elaborated using examples from file sharing applications, and successfully evaluated by implementing a prototype service-based file sharing grid.

Keywords Trust · Recommendation · Interaction · Service-based software

This research is partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

M. G. Uddin
Department of Electrical Engineering and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada

M. Zulkernine (✉)
School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada
e-mail: mzulker@cs.queensu.ca

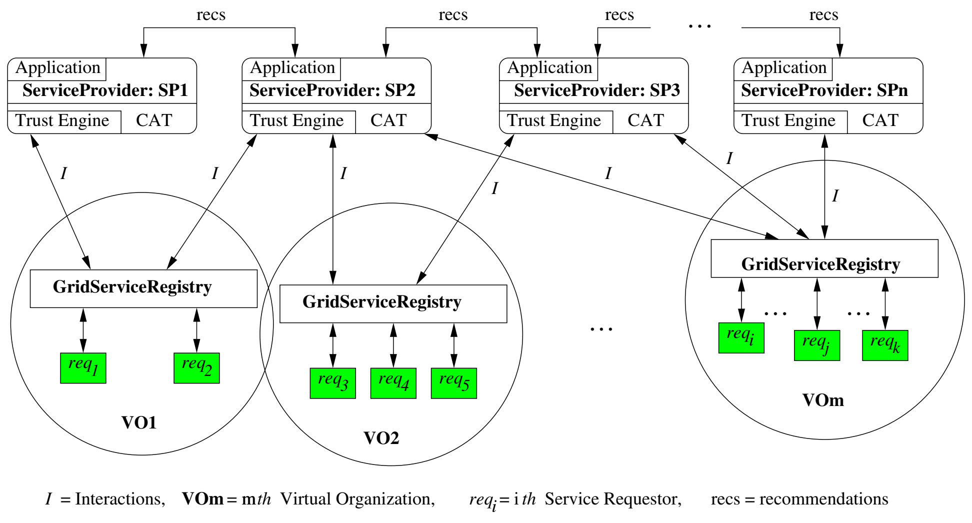
S. I. Ahamed
Department of Mathematics, Statistics and Computer Science, Marquette University, Wisconsin, Milwaukee, WI 53233, USA

1 Introduction

Mutual collaboration is necessary in open and dynamic distributed service-based systems, where entities rely on each other for achieving goals, utilizing resources, and performing tasks. Entities are open to collaborate with any other entities, introducing dynamism by joining and leaving the system at any time. Service providers collaborate with requestors through the provisioning of services, and with other providers through the exchanging of recommendations about requestors. They exchange recommendations to convey their opinions on requestors. However, service providers are exposed to requestors scattered across multiple organizational domains, so can be exploited by untrustworthy requestors while providing services [1].[1] Given that, provider software need to

[1] For brevity, we use provider to denote service provider and requestor to denote service requestor.

Fig. 1 A grid structure incorporating trust-aware service provider



automatically decide on which entity they should trust for service collaboration [2].

Trust helps in dynamic decision making by covering a number of quality attributes (e.g., security). "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" [3]. A trusting agent is called a trustor entity, and a trusted agent is called a trustee. A context describes a situation that influences a trust relationship. In service-based software systems, a service provider builds trust relationships with a service requestor based on the analysis of service-based interactions. An interaction represents collaboration between a service provider and a requestor at the service usage level [4]. In this case, the services can be considered as contexts [5]. A trust-aware provider analyzes service-based interactions from trust perspectives and makes decisions accordingly [6].

Although much research have been devoted to develop systems for trust-based access control and decision making [2,5,7–16], secret key sharing [17], or appropriate service provider selection [6], to date, no mechanism has been proposed to automate service collaboration for analyzing the trustworthiness of service requestors at run-time. In this paper, we quantify service collaboration from trust perspectives using our trust model called CAT (Context-Aware Trust) [18], and dynamically decide on service requests at run-time based on our proposed trust-based service collaboration algorithm. We extend the interaction-based trust properties of [18] to service-based systems. The trustworthiness of requestors is calculated by comparing service-based interactions against trust rule(s). A trust rule encapsulates a trust scenario that describes a particular trust relationship [19]. The rules are risk-aware and the calculation schemes are context-aware. A context-similarity parameter is introduced to measure similarity between services. A time-based ageing parameter is used to reduce trust values over time, when no further interaction takes place between entities. Recommendations from closer entities (i.e., neighbors) are considered as direct, while a path-based ageing parameter is applied for deriving indirect recommendations. An accuracy mechanism is described to differentiate between reliable and unreliable recommendations. The proposed trust-based service collaboration algorithm operates on service requests and uses CAT to decide whether the corresponding requestors are trustworthy enough for the requested services. The approach is illustrated using examples from file sharing grid, implemented in a prototype service-based grid system, and successfully evaluated.

The rest of the paper is organized as follows. Section 2 sets the stage of the service collaboration approach by defining the type of trust-aware service-based systems considered in this paper. Section 3 presents some well-known collaboration-based trust properties that are deemed necessary for the proposed systems. The trust model is presented in Sect. 4, and the trust-based service collaboration algorithm is described in Sect. 5. The implementation and experimental evaluation is discussed in Sect. 6. Section 7 reviews the related collaboration-based trust models. The work is concluded in Sect. 8 with a few future research directions.

2 Overview of trust-based service collaboration

In a service-based grid system, service providers are connected to a number of organizations through which requestors can access the services. Figure 1 presents an example system structure, having each provider incorporating CAT as a trust engine to analyze service-based interactions from trust perspectives. Service requestors are scattered across multiple organizational domains, with each domain representing a

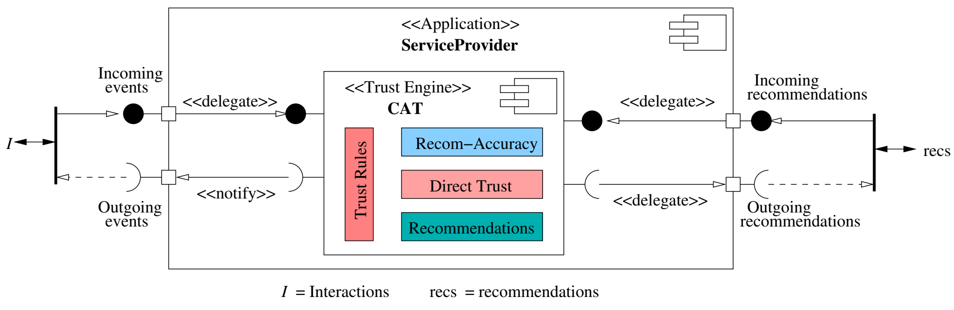


Fig. 2 A system architecture incorporating CAT

virtual organization [20]. A provider is registered to an organization through the corresponding `GridServiceRegistry` of the organization. An organization can subscribe to a number of service providers, as well as a service provider can be registered to a number of organizations. Requestors use services by interacting with providers. Moreover, providers can interact with each other to exchange recommendations between them.

The trust engine, CAT is based on service collaboration between entities by considering that interactions have certain relevant properties from the perspectives of trust. An interaction can have a number of trust rules. Based on the trust rules, interactions are analyzed at run-time. Upon the outcome of an interaction, the interacting entities judge each other. A service requestor is penalized for the violation of a trust rule and awarded for no such violations. Direct trust is derived from the outcomes, which is updated after each interaction. Recommendations are collected and compared against the outcome of the corresponding interactions. This comparison follows an accuracy mechanism to differentiate between reliable and unreliable recommendations. Figure 2 places CAT as a trust-based decision making component («`Trust Engine`») in a service providing software («`Application`»). CAT has four repositories, `Trust Rules`, `Direct Trust`, `Recommendations`, and `Recom-Accuracy`. Interactions (I) are monitored using rules from the `Trust Rules` repository. The `Direct Trust` repository is updated after each interaction. Recommendations are stored in the `Recommendations` repository. The `Recom-Accuracy` repository records and updates the accuracy of different recommendations after each interaction. Service requests (i.e., `Incoming events`) are delegated to CAT and analyzed using the corresponding trust rules. The trust engine makes service-based decisions («`notify`») based on direct trust and recommendations (i.e., `recs`) that are sent to the corresponding service requestor (i.e., `Outgoing events`). Section 4 discusses the different calculation schemes of CAT, and Sect. 5 presents the algorithm for automatic decision making.

According to Tie-Yan et al. [21], a "session" (process) - level trust model in service-based grid system has two steps. In the first step, a sub-process is used to analyze the identity of the requestor among the virtual organizations that register the service providers and connect the providers to the requestors. In the second step, another sub-process is used in the grid domain to analyze the confidence on the requestors to grant them the requested services. The analysis of confidence on the requestors is performed based on the quantification of the trustworthiness of the requestors. The second step can be performed either in a central trust architecture that will monitor all the interactions between the providers and the requestors, or in trust architectures that reside in each provider. In our proposed trust-based service collaboration approach, we have focused on the second sub-process based on the implementation of a trust model in each service provider by assuming that the identity of a requestor has already been verified by the corresponding grid service registry.

3 Collaboration-based trust properties

Since the trust-based analysis of service collaboration requires the analysis of interaction between entities, we develop CAT as an interaction-based trust model, following well-known interaction-based trust properties specified in [5,9,14,15,22–24] and listed below.[2]

P1. Context-awareness (CA): Trust of entity $E1$ on entity $E2$ for context c_i (i.e., service) at time t ($T(E1, E2, c_i, t)$) does not imply the same trust for another context c_j.

$$T(E1, E2, c_i, t) \nRightarrow T(E1, E2, c_j, t), \quad \text{where } i \neq j$$

P2. Rule-oriented (RO): An interaction I can have a number of trust rules (r) to analyze its outcomes. The total outcome O of I is a function of all the outcomes as measured by the corresponding trust-rules ($r_1, r_2, \ldots, r_n$).

$$O(I) = f(O(r_1), O(r_2), \ldots, O(r_i), \ldots, O(r_n))$$

P3. Risk-awareness (RA): For each interaction, the corresponding trust rules capture the possible risks associated with

[2] For the sake of simplicity, we denote both the service provider and the requestor as entities (E). However, the trustor and the recommender entities are always service providers, and the trustee entities are always clients (i.e., service requestors). A context (c_i) always denotes a service.

it. The higher the risk of an outcome, the higher is the importance level assigned to the corresponding trust rule.

$$Risk(O(r_i)) > Risk(O(r_j)) \Rightarrow (Imp(r_i) > Imp(r_j)),\ \text{where } i \neq j$$

P4. Recommendation-based (RB): The total trust of $E1$ on $E2$ can be a function of the direct trust ($T_D(E1, E2, c_i, t)$) and the total recommendation trust of $E1$ on $E2(R(E1, E2, c_i, t))$, where the total recommendation trust is calculated from the recommendations of other entities to $E1$ on $E2$. However, an entity can discard recommendations in calculating total trust which makes the following condition not applicable to all situations.

$$\exists c_i (T(E1, E2, c_i, t) = f(T_D(E1, E2, c_i, t), R(E1, E2, c_i, t)))$$

P5. Recommendation-filtration (RF): An entity can impose a certain accuracy threshold to differentiate between reliable and unreliable recommendations. The accuracy (A) of a recommendation can be determined by comparing it against the corresponding direct trust, obtained by the analysis of direct interactions with the corresponding trustee (i.e., recommended entity) [25]. A recommendation from $E3$ to $E1$ about $E2$ should be discarded, if the recommendation accuracy of $E3$ for this context are below the accuracy threshold (τ_R).

$$\forall c_i (UnReliable(R(E3, E2, E1, c_i, t)) \Rightarrow (A(E3, E2, E1, c_i, t) < \tau_R(c_i)))$$

P6. Semi-transitive (ST): *"Trust cannot be trivially propagated"* [9]. If $E1$ trusts $E2$ and $E2$ trusts $E3$, that cannot be concluded as $E1$ will trust $E3$. However, if $E2$ recommends $E3$ to $E1$, then there may be a partial trust relationship between $E1$ and $E3$.

$$\forall c_i (T(E1, E3, c_i, t) \not\Rightarrow (T(E1, E2, c_i, t), T(E2, E3, c_i, t)))$$
$$\exists c_i (T(E1, E3, c_i, t) \Rightarrow (T(E1, E2, c_i, t), R(E2, E1, E3, c_i, t)))$$

P7. Non-symmetric (NS): For any context c_i, trust of $E1$ on $E2$ at time t does not imply the same trust of $E2$ of $E1$ for c_i.

$$\forall c_i (T(E1, E2, c_i, t) \not\Rightarrow T(E2, E1, c_i, t))$$

P8. Dynamic (D): The trust value of $E1$ on $E2$ changes over time due to new interactions.

$$\forall c_i (T(E1, E2, c_i, t) \neq T(E1, E2, c_i, t + \Delta t))$$

P9. Time-based ageing (TA): Without any further interaction, the trust value of $E1$ on $E2$ decreases over time.

$$\forall c_i (T(E1, E2, c_i, t) > T(E1, E2, c_i, t + \Delta t))$$

P10. Path-based ageing (PA): Recommendations from nearer entities are given more importance than the recommendations from the entities which are more far away.

P11. Sample-based (SB): The total trust of a trustor on a trustee at a particular time is based on all the previous direct trust values (i.e., samples) between them. The total trust of $E1$ on $E2$ at time t combines all the direct trusts obtained for c_i until time t.

$$\forall c_i (T(E1, E2, c_i, t) = f(T_D(E1, E2, c_i, t), T_D(E1, E2, c_i, t - 1), \ldots))$$

P12. Closeness-oriented (CO): The trust on a trustee for a particular service can be obtained from similar trust from other services based on their closeness with each other. Therefore, the trust on a trustee for a context (c_i), can be influenced by the monitored values of similar other trust rules related to c_j on the same trustee, where c_i and c_j are close to a certain degree.

$$\exists \{c_i, c_j\} (T(E1, E2, c_i, t) = f(T(E1, E2, c_j, t)),\ \text{where, } i \neq j.$$

P13. Variance-based (VB): The degree to which the trust of trustor ($E1$) about a trustee ($E2$) varies from the desired value should be considered to authorize the trustee to particular service (c_i). This is related to the interaction threshold (τ_I) that is used to make decision about an interaction with a service requestor

$$\forall c_i (Authorize(E1, E2, c_i, t) \Rightarrow (T(E1, E2, c_i, t) \geq \tau_I(c_i)))$$

4 CAT: a trust model for quantifying collaboration

Service collaboration is quantified using the trust model, CAT by analyzing interactions with requestors from trust perspectives, and calculating the trustworthiness of requestors based on direct observations and recommendations. We provide the detailed calculation schemes of CAT in the next few subsections.

4.1 Building collaboration-based confidence on service requestors

'Confidence' is a quantified satisfaction obtained by a trustor on a trustee from the analysis of a service-based interaction between them. The calculation of trust begins whenever an interaction takes place, its outcome being used to calculate the confidence in the interaction. The calculation of confidence is thus the first step of the direct trust calculation (see Sect. 4.2). Each interaction has certain trust rule(s) that analyze the outcomes of the interaction at run-time. Each of the rules is assigned a particular trust value. For example, a trust rule can be assigned a high (H), medium (M), or low (L) trust value. The satisfaction of each trust rule awards the trustee a certain level of belief, while the violation of each trust rule penalizes the trustee with a certain level of disbelief. The highest belief is assigned a value of 1, medium belief 0.8, low belief 0.6, while the highest disbelief score is 0, medium disbelief is 0.2, and low disbelief is 0.4. A neutral value of 0.5 denotes no belief and no disbelief. The total belief (I_b) of trustor $E1$ on trustee $E2$ for context c_i at time t from interaction I is calculated using Eq. 1, where $B(rn)$ contains the belief value of the trust rule indexed as rn, n_b is the total number of trust rules with belief outcomes. Similarly, total disbelief I_d (range [0,1]) is calculated using Eq. 2, where $D(rn)$ contains the disbelief value of the trust rule indexed as rn, and n_d is the total number of trust rules with disbelief outcomes. The confidence (μ) of $E1$ on $E2$ from I about context c_i at time t is calculated from I_b and I_d using Eq. 3. The calculation of μ thus expresses the level of confidence $E1$ gathers on $E2$ from a particular interaction I at an interaction time t.

$$I_b(E1, E2, c_i, t) = \frac{b(E1, E2, c_i, t)}{n_b}, \quad \text{where } b(E1, E2, c_i, t) = \sum_{rn=0}^{n_b} B(rn). \tag{1}$$

$$I_d(E1, E2, c_i, t) = \frac{d(E1, E2, c_i, t)}{n_d}, \quad \text{where } d(E1, E2, c_i, t) = \sum_{rn=0}^{n_d} D(rn). \tag{2}$$

$$\mu(E1, E2, c_i, t) = w_b I_b(E1, E2, c_i, t) + w_d I_d(E1, E2, c_i, t), \quad where\ w_b + w_d = 1. \tag{3}$$

In Eq. 3, w_b and w_d (range [0,1]) are weights assigned to I_b and I_d respectively. In situations when the trustors want to emphasize the interactions with trustees despite their misbehavior, they can give more importance to w_b. By putting more weight on w_b, trustors can ensure that trustees can have additional chances up to certain limit, even though they misbehaved in the past. For example, if we provide values as $w_b = 0.8$, and $w_d = 0.2$, then we consider that the satisfactory interactions with a trustee (calculated by I_b) are given four times more importance than the unsatisfactory interactions (calculated by I_d), and so the trustee may have additional chances even after it has misbehaved earlier.

The calculation of confidence uses trust rules to analyze the trustworthiness of entities. Based on the graveness of the trust violation, the importance value of trust rules differ. For example, a trust rule considering a highly malicious outcome is provided with a high importance. The use of trust rules in the trust calculation makes the trust model rule-oriented (see property P2 of Sect. 3), and these calculations are performed for each of the provided services which makes CAT context-aware (see property P1 of Sect. 3). Moreover, each trust rule is assigned a particular importance value based on its probable risk outcome. This satisfies the risk-awareness property (see property P3 of Sect. 3).

4.2 Calculating direct trust on service requestors

Direct trust (T_D) of an entity $E1$ on $E2$ for context c_i at time t is calculated based on the analysis of the corresponding interactions between them. The value of T_D changes after each interaction based on the outcome of the interaction, satisfying the dynamism of trust (see property P8 of Sect. 3). The calculation of direct trust is the most important, as it is based on self-observation and does not depend on recommendations. The calculation of a direct trust T_D of $E1$ on $E2$ for c_i at time t has two options. The first option calculates T_D using the confidence (μ) related to c_i, while the second option calculates T_D based on another context c_j, which is similar to c_i to some extent.

The first option of direct trust calculation is performed using Eq. 4, where δ (range [0,1]) is the weighting factor. When δ is given less weight (e.g., for a fast changing environment), the latest confidence is preferred more than previous confidences. Upon the calculation of confidence μ for a context c_i at time t, the previous direct trust ($T_D(E1, E2, c_i, t_o)$) is retrieved from the corresponding interaction records, and then new direct trust is calculated.

$$T_D(E1, E2, c_i, t) = \delta T_D(E1, E2, c_i, t_o) + (1-\delta)\mu(E1, E2, c_i, t). \tag{4}$$

The second option of direct trust calculation is used only when a decision needs to be made for $E2$ about c_i and there is no record of direct interaction for that context. Equation 5 handles this problem by calculating direct trust T_D of $E1$ on $E2$ for context c_i at time t using $\Re(c_i, c_j)$ (range [0, 1]), a context-similarity parameter which measures the similarity between context c_i and c_j, and $T_D(E1, E2, c_j, t)$ which provides the direct trust of $E1$ on $E2$ for c_j. The context-similarity parameter is discussed further in Sect. 4.2.1.

$$T_D(E1, E2, c_i, t) = T_D(E1, E2, c_j, t)\Re(c_i, c_j). \tag{5}$$

Note that in Eqs. 4 and 5, t represents the current time index, and t_o represents the time of the most recent direct trust calculation stored for a particular entity pair and context.

Without further interaction, direct trust decreases over time. For example, if $E1$ calculates direct trust T_D on $E2$ for c_i at time t_o and the recent time is t, then the recent direct trust $T_{D_r}(E1, E2, c_i, t)$ should decrease without further interaction due to the time-based ageing property (see property P9 of Sect. 3). Equation 6 calculates the recent direct trust of $E1$ on $E2$ about c_i at time t, where we assume that the direct trust T_D of $E1$ on $E2$ about c_i at time t_o is obtained from the corresponding interaction records (i.e., the T_D previously calculated using Eq. 4 or Eq. 5). For example, suppose we have a record of direct trust at time $t_o = 100$ but at time $t = 200$, we need the direct trust from the record. To obtain the recent direct trust, we use Eq. 6 on the direct trust T_D found at time $t_o = 100$. The calculation of recent direct trust is performed using a time-based ageing parameter $\gamma(t_o, t, c_i)$ (range [0, 1]). The time-based ageing parameter is elaborated further in Sect. 4.2.2.

$$T_{D_r}(E1, E2, c_i, t) = T_D(E1, E2, c_i, t_o)\gamma(t_o, t, c_i). \qquad (6)$$

4.2.1 *Context-similarity parameter ($\Re(c_i, c_j)$).*

Recall that, the context-similarity parameter is necessary, when no previous interaction value is available in the record database for a context. We assume that every context has some keywords to describe it, and $\kappa(c_i)$ denotes the total set of keywords describing context c_i. Then a similarity metric can be based on the ratio of keywords two contexts (e.g., c_i and c_j) have in common. Equation 7 provides the definition of context-similarity parameter, where the numerator takes total number of similar keywords that two contexts c_i and c_j have in common (i.e., the union of all keywords for c_i and c_j), while the denominator takes total number of distinct keywords used to describe c_i and c_j (i.e., the intersection of all keywords for c_i and c_j).

$$\Re(c_i, c_j) = \frac{length\_of[\kappa(c_i) \cap \kappa(c_j)]}{length\_of[\kappa(c_i) \cup \kappa(c_j)]}. \qquad (7)$$

For example, suppose a file-server application has three types of services (i.e., contexts): `UploadPDFFile` with keywords `{write, pdf, file}`, `UploadDocFile` with keywords `{write, doc, file}`, `Login` with keywords `{userName, passWD}`. Therefore, the numerator is 0 and the denominator is 5 for `Login` and `UploadPDFFile`, giving the value of $\Re(c_i, c_j)$ as 0 (i.e., the contexts are not similar at all). However, for `UploadPDFFile` and `UploadDocFile` the numerator is 2, denominator is 4, and so the value of the parameter is 0.50 (i.e., if $E1$ trusts $E2$ for uploading `doc` files, it can reasonably trust $E2$ for uploading `PDF` files). This parameter helps in making decisions in similar situations, so as to decrease the reliance on recommendations and to rely on self-observation more. The use of service-similarity satisfies the closeness-orientedness property (see property P12 of Sect. 3).

The notion of context-similarity is first provided by Toivonen et al. [26] and then Billhardt et al. [10], where the first ones consider a context-tree, with the similarity between contexts is calculated as the distance between them based on the root node of the tree. The problem with the above approach is the context-tree, which might not be present for all the contexts in the system. However, if we can construct such a tree based on our service-keywords, our parameter will produce the same result. Billhardt et al. [10] considers a chain of services in an organizational structure, where the invocation of one service may lead to another service. Since we apply keywords to describe each service, our approach is applicable to services in general, and it eliminates the need for any such context-tree or organization-level service hierarchy. Although our approach of context-similarity can be adapted for a hierarchical structure, our parameter will produce different result than [10].

4.2.2 *Time-based ageing parameter ($\gamma(t, t_r, c_i)$).*

This parameter is used to reduce a trust value over time when no further interaction takes place. It is calculated using Eq. 8, where Υ (range [0, 1]) is a time-based ageing factor. The closer Υ is to 1, the lower the value of γ, and the value of previous direct trust decreases with the decrease in γ. For essentially static environments, where very few number of interactions take place within a particular time period, the value of Υ might be chosen close to 0.

$$\gamma(t_o, t, c_i) = 1 - \frac{(t - t_o)\Upsilon}{t}. \qquad (8)$$

In Eq. 8, t_o is the last time of interaction between entity $E1$ and $E2$ for c_i, and t is the current time, when a trust calculation on $E2$ for c_i is required. $(t - t_o)/t$ provides the difference between t and t_o in terms of t, where the division by t restrains the ratio in the range [0, 1].

The notion of time-based ageing is first introduced by Sabater and Sierra in their trust model called REGRET [14], calculated as $f(t_o, t) = \frac{t_o}{t}$. The problem with this parameter is that it only takes the ratio of current time and old time, and does not give preference to the system environment. For example, in a slowly interacting system, entities do not interact frequently, and so this parameter may drastically reduce the recent direct trust value. To address this shortcoming, we use our time-based ageing factor (Υ), where the value of Υ is chosen based on the nature of the system environment. Unlike them, we are taking the difference between the old time and the recent time, so we subtract our ratio from 1.

4.3 Calculating recommendation and recommendation accuracies

We consider two types of recommendations: direct and indirect. Recommendations obtained from immediate neighbors [5] are considered to be direct, while the recommendations obtained from further afield are considered indirect. Figure 3 describes the two types of recommendations: the recommendation from $E5$ on $E2$ to $E1$ is indirect as it comes through $E4$ (i.e., path-length = 3), the recommendation from

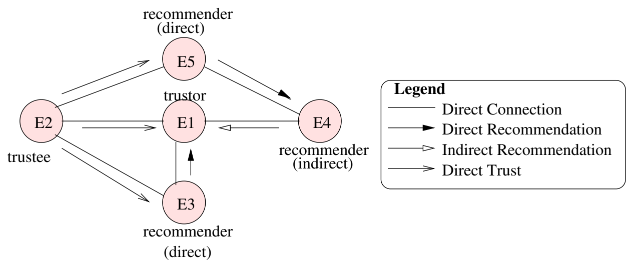


Fig. 3 Direct and indirect recommendations in CAT

$E3$ on $E2$ to $E1$ is direct (i.e., path-length = 2). We consider the number of visited nodes (i.e., vertices) in a recommendation network to calculate a path-length, where a recommendation network consists of all the providers in a service-based system. A recommendation path with path-length more than 2 is indirect.

An entity $E3$ provides direct recommendation (R_D) to $E1$ for $E2$ about context c_i at time t using Eq. 9. η (range [1, 0]) is the weighting factor an entity imposes on its direct trust calculated at time t_o for the purpose of recommendation. A recommendation value R_D is at most equal to the corresponding direct trust value in our system.

$$R_D(E3, E1, E2, c_i, t) = \eta T_D(E3, E2, c_i, t_o). \tag{9}$$

We do not consider context-similarity parameter in providing recommendations, and so a recommender in our system only recommends a trustee for context c_i if the trustee has previously interacted with the recommender for the same context c_i. The reason for not choosing context-similarity is that this parameter only provides an assumed trust value derived from a similar context. Since entities in our system judge the recommendations from each other very carefully (i.e., through the calculation of recommendation accuracy), they may want to avoid any more uncertainty in providing or receiving a recommendation.

The indirect recommendation (R_I) from entity $E5$ to $E1$ on $E2$ about c_i at time t is calculated using Eq. 10, where $\vartheta(M, \Lambda, c_i)$ is the path-based ageing parameter. $\vartheta(M, \Lambda, c_i)$ is adapted from [5], but it is calculated in a different manner using Eq. 11. M is the visited path-length in a recommendation path, Λ is the maximum allowed path-length for this recommendation path, and Ψ is the distance-based ageing factor. Using path-based ageing parameter, indirect recommendations are given less weight than direct recommendations.

$$R_I(E5, E1, E2, c_i, t) = R_D(E5, E1, E2, c_i, t)\vartheta(M, \Lambda, c_i). \tag{10}$$

$$\vartheta(M, \Lambda, c_i) = 1 - \frac{(M-2)\Psi}{\Lambda}, \quad \text{where } M \geq 2. \tag{11}$$

The value of Λ can be changed based on preferences. For example, in a highly uncertain service-based system, where a service provider has no way of knowing about other recommenders, it can consider recommendations only from a number of recommenders whom it or its neighbors know properly. Moreover, it takes considerable time to get a recommendation reply if the maximum allowable path length is pretty long. Therefore, if a service provider wants to rely more on the recommenders closer to it as well as wants to avoid delay in providing services it can set a low value for Λ. The use of a path-based ageing parameter satisfies the path-based ageing property (see property P10 of Sect. 3).

The original parameter proposed by [5] is $1 - \frac{(M-1)\Psi}{10}$. The problem with the original parameter is that it has 10 as denominator, and therefore, at some point it may become negative. For example, for $M = 16$, $\Psi = 0.9$, the value of the parameter is -0.35. We address this problem by using maximum allowed path length Λ instead of 10. Moreover, since they consider the edges in a recommendation network to calculate path-length as opposed to the vertices, the subtraction in their numerator is by 1, while it is by 2 in our numerator.

We address the problem of incomplete network knowledge in path-based ageing by adopting the following two measures. First, if there is no recommendation found within the path length Λ, the provider to whom the request has been forwarded, reply to the recommendation requestor by sending a reply of undefined recommendation. Second, if there is no recommender found before the maximum allowed path length, the corresponding recommendation is also treated as undefined.

The accuracy (A) of $E3$ to $E1$ in providing a direct recommendation for $E2$ about context c_i is calculated using Eq. 12, where $\Delta R_D(E3, E1, E2, c_i, t)$ calculates the absolute difference between the provided recommendation and the calculated direct trust (see Eq. 13).

$$A_{R_D}(E3, E1, E2, c_i, t) = 1 - \Delta R_D(E3, E1, E2, c_i, t). \tag{12}$$

$$\Delta R_D(E3, E1, E2, c_i, t) = |R_D(E3, E1, E2, c_i, t) - T_D(E1, E2, c_i, t)|. \tag{13}$$

Each trustor keeps an accuracy table (AT), where it updates the accuracy of every recommendation after the analysis of the corresponding interaction with a trustee. The accuracy of a direct recommender $E3$ to $E1$ about $E2$ regarding context c_i at time t in the AT is denoted by $AT_{R_D}(E3, E1, E2, c_i, t)$. The update in the AT at time t is performed using Eq. 14 by considering previous recommendation accuracy ($AT_{R_D}(E3, E1, E2, c_i, t_o)$) at time t_o and new recommendation accuracy ($A_{R_D}(E3, E1, E2, c_i, t)$) at time t. ζ (range [1, 0]) weights the importance of the previous accuracy and the current accuracy of a recommender. This is necessary, since an entity may be completely wrong at some point. However, this cannot be used to conclude on all of its recommendations.

$$AT_{R_D}(E3, E1, E2, c_i, t) = \zeta AT_{R_D}(E3, E1, E2, c_i, t_o) + (1-\zeta)A_{R_D}(E3, E1, E2, c_i, t). \quad (14)$$

Using Eqs. 12 and 14, unreliable recommendations can be detected. A recommender is considered most reliable with accuracy 1 and most unreliable with accuracy 0. An entity can impose a certain accuracy threshold to discard recommendations. This satisfies the recommendation filtration property (see property P5 of Sect. 3).

The calculation of recommendation accuracy in Eqs. 12–14 closely follows the calculation of Azzedin and Maheswaran [25], and like our Eq. 13, they also measure the difference between direct trust and provided recommendations. However, they do not differentiate between direct and indirect recommendations. Moreover, Eqs. 12 and 14 are context-aware in our system, as opposed to the generalized forms used in their system.

Equations 12–14 are also used for calculating the accuracy of indirect recommendations by replacing R_D by R_I. The calculation of recommendation points out that an entity can be trusted based on the provided recommendations to a certain extent. This satisfies the semi-transitive and recommendation-based properties (see properties P4 and P6 of Sect. 3).

4.4 Calculating total recommendation trust

A direct/indirect recommendation trust (θ) is calculated by binding a direct/indirect recommendation with its previous accuracy. The direct/indirect recommendation trust as measured by $E1$ on $E2$ based on the recommendation provided by $E3$ for context c_i at time t is calculated by Eq. 15. The total recommendation trust of $E1$ on $E2$ about c_i is calculated using Eq. 16, where N is the total number of direct and indirect recommenders.

$$\theta_R(E3, E1, E2, c_i, t) = R(E3, E1, E2, c_i, t)AT_R \times (E3, E1, E2, c_i, t).$$

where $R = R_D$ for path-length $= 2$ and $R = R_I$ for path-length > 2. (15)

$$R(E1, E2, c_i, t) = \frac{\sum_{e=0}^{N} \theta_R(e, E1, E2, c_i, t)}{N}. \quad (16)$$

4.5 Calculating total trust

After obtaining the direct trust and the total recommendation trust on $E2$, $E1$ calculates total trust (T) on $E2$ for context c_i using Eq. 17. The calculation is performed based on recent direct trust T_{D_r} obtained by Eq. 6, and total recommendation trust obtained from Eq. 16. α (range [0, 1]) is the self-confidence level. An entity can increase α to rely on self-observation more, and in the limit can discard all recommendations by using $\alpha = 1$.

$$T(E1, E2, c_i, t) = \alpha T_{D_r}(E1, E2, c_i, t) + (1-\alpha)R(E1, E2, c_i, t). \quad (17)$$

5 Trust-based collaborative dynamic decision making

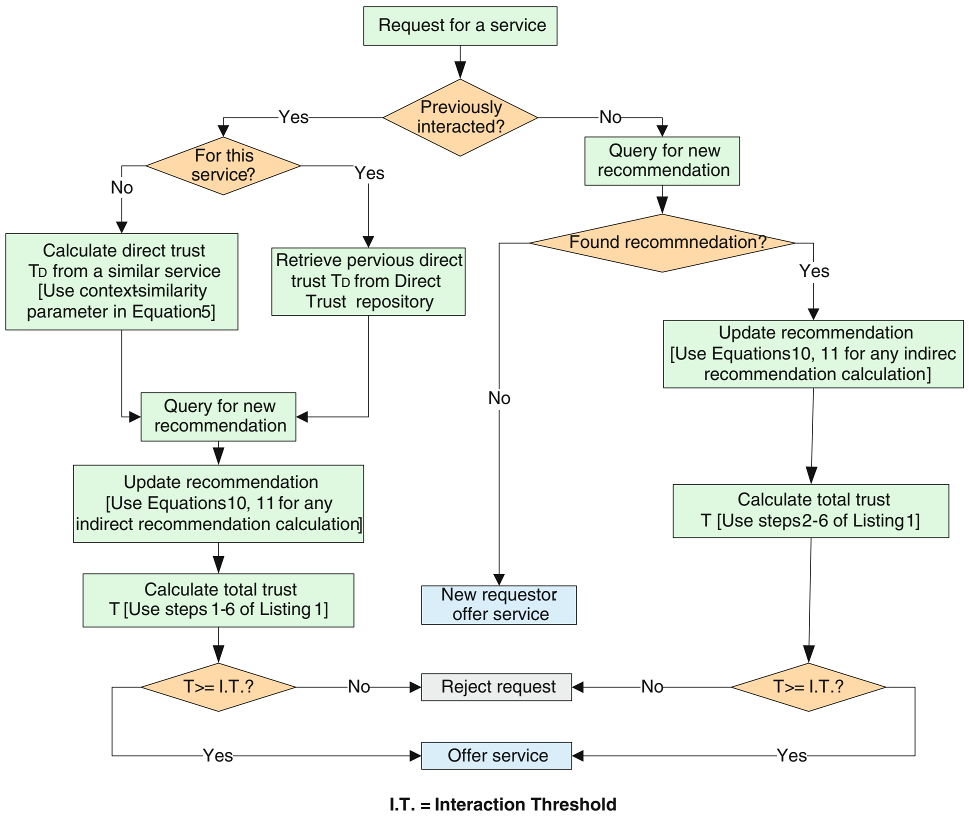
The quantification of trust helps in making dynamic decision using our trust-based service collaboration algorithm which is used to accept or reject service requests (see Sect. 5.1). The overall approach is elaborated using different file sharing scenarios (see Sect. 5.2).

5.1 A trust-based service collaboration algorithm

In this section, we present a trust-based service collaboration algorithm which uses CAT to make trust-based dynamic decision on behalf of a service provider (sp). Based on a request for a service (s) from a service requestor (sr), the algorithm provides a decision on whether the requestor will be granted the service or not. We assume that the time of this service request is t. A flowchart representation of the approach is provided in Algorithm 1 that operates in two distinct conditions:

- The requestor has interacted with the provider previously, and therefore, it is known to the provider.
- The requestor has not interacted with the provider before, and therefore, it is unknown to the provider.

The provider uses CAT to make decisions, checking whether the requestor has interacted previously for the requested service. This is determined by searching the `Direct Trust` repository for available interaction record(s) of the requestor. If the requestor has a previous interaction record for the service, the direct trust value on the requestor about the service is loaded from the `Direct Trust` repository. However, in case of no direct interaction with the requestor for that particular service, the provider uses the context-similarity parameter of Eq. 7 in Eq. 5 to obtain the direct trust value on the requestor for the service (i.e., the second option of calculating T_D). Then the provider makes queries for recommendations about the requestor for the service. Upon the receipt of all recommendations, the provider updates the `Recommendations` repository, using Eq. 10 and Eq. 11 for any indirect recommendation. Finally, the total trust T is calculated by following all the steps in Listing 1, where the accuracy threshold τ_A is set up by the provider. The accuracy threshold can


Request for a service
Previously interacted?
Yes
No
For this service?
No
Yes
Query for new recommendation
Found recommnedation?
Calculate direct trust TD from a similar service [Use context-similarity parameter in Equation5]
Retrieve pervious direct trust TD from Direct Trust repository
Yes
Update recommendation [Use Equations10, 11 for any indirec recommendation calculation]
No
Query for new recommendation
Update recommendation [Use Equations10, 11 for any indirect recommendation calculation]
Calculate total trust T [Use steps2-6 of Listing1]
Calculate total trust T [Use steps 1-6 of Listing 1]
New requestor offer service
T>= I.T.?
No
Reject request
No
T>= I.T.?
Yes
Offer service
Yes
I.T. = Interaction Threshold

Algorithm 1 Trust-based service collaboration

Listing 1 Total trust calculation in Algorithm 1

1. Use Eq. 6 to obtain recent direct trust, $T_{D_r}(sp, sr, s, t)$.
2. Load recommendations about sr for s from the `Recommendations` repository.
3. Load accuracy of each recommender from the `Recom-Accuracy` repository.
4. Discard recommendations with accuracy level less than accuracy-threshold, τ_A.
5. Calculate total recommendation trust $R(sp, sr, s, t)$ using Eqs. 15 and 16.
6. Calculate total trust $T(sp, sr, s, t)$ using Eq. 17.

vary for different service providers. The service is offered to the requestor if T is not less than the interaction-threshold ($I.T.$) for the service. The interaction threshold is set up by the sp and can vary for different service providers.

When the provider finds no record for the requestor in its `Direct Trust` repository, it queries for recommendations about the requestor for the service. If no recommendations are found, provider determines that the requestor is an absolutely new entity to the system. It then assigns the requestor the neutral value 0.5 and offers service. However, the entity is not considered completely new if some recommendations are found. In this case, the `Recommendations` repository is updated upon the arrival of all recommendations. Then the total trust T is calculated using Steps 2 to 6 of Listing 1, and the service is provided if warranted (i.e., $T \geq I.T.$).

The use of interaction threshold to decide about a service request satisfies the variance-based property of collaboration-based trust (see property P13 of Sect. 3). After the service is granted to the requestor, an interaction session for this service is initiated between the provider and the requestor, during which time the interaction events are analyzed using the corresponding trust rules. After the analysis of each interaction based on the trust rules, the provider calculates μ and updates T_D in the `Direct Trust` repository using Eqs. 1–4. The provider then calculates the corresponding recommendation accuracies using Eqs. 12–14, updating the `Recom-Accuracy` repository.

5.2 An illustrative example

In this section, we present examples from file sharing applications to make trust-based automatic decisions using the proposed algorithm. We adapt the system structure of Fig. 1 into a file storage grid [7], having a number of servers to provide file sharing services (i.e., file upload, open, and search) that can be used for collaborative groupware projects, or scientific purposes [20]. However, file servers need to reason about the trustworthiness of requestors before granting them access to

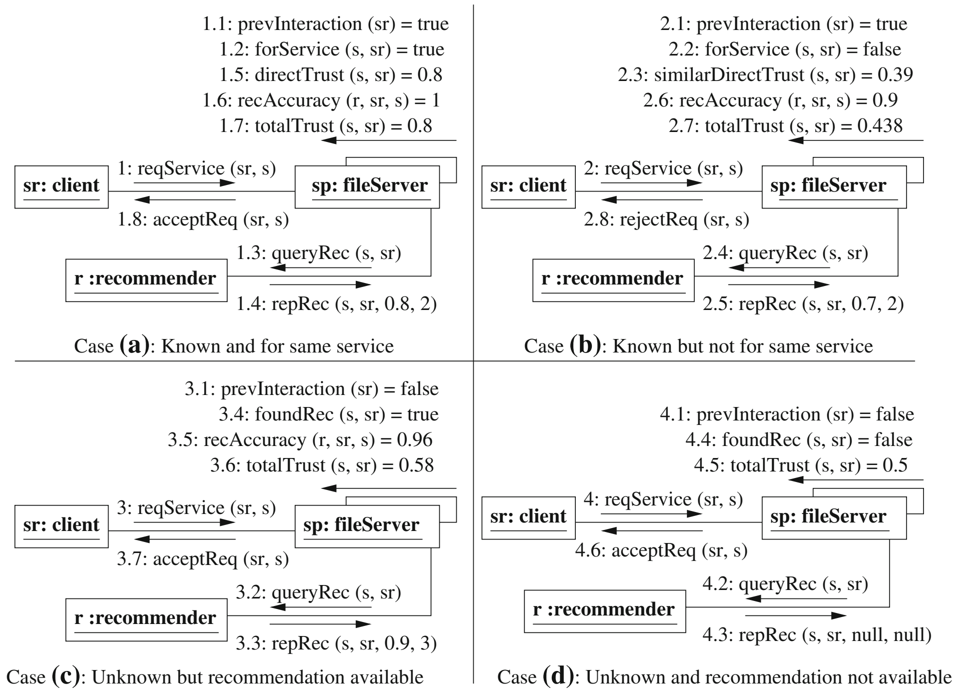

1.1: prevInteraction (sr) = true
1.2: forService (s, sr) = true
1.5: directTrust (s, sr) = 0.8
1.6: recAccuracy (r, sr, s) = 1
1.7: totalTrust (s, sr) = 0.8
sr: client
1: reqService (sr, s)
sp: fileServer
1.8: acceptReq (sr, s)
1.3: queryRec (s, sr)
r :recommender
1.4: repRec (s, sr, 0.8, 2)
Case (a): Known and for same service
2.1: prevInteraction (sr) = true
2.2: forService (s, sr) = false
2.3: similarDirectTrust (s, sr) = 0.39
2.6: recAccuracy (r, sr, s) = 0.9
2.7: totalTrust (s, sr) = 0.438
sr: client
2: reqService (sr, s)
sp: fileServer
2.8: rejectReq (sr, s)
2.4: queryRec (s, sr)
r :recommender
2.5: repRec (s, sr, 0.7, 2)
Case (b): Known but not for same service
3.1: prevInteraction (sr) = false
3.4: foundRec (s, sr) = true
3.5: recAccuracy (r, sr, s) = 0.96
3.6: totalTrust (s, sr) = 0.58
sr: client
3: reqService (sr, s)
sp: fileServer
3.7: acceptReq (sr, s)
3.2: queryRec (s, sr)
r :recommender
3.3: repRec (s, sr, 0.9, 3)
Case (c): Unknown but recommendation available
4.1: prevInteraction (sr) = false
4.4: foundRec (s, sr) = false
4.5: totalTrust (s, sr) = 0.5
sr: client
4: reqService (sr, s)
sp: fileServer
4.6: acceptReq (sr, s)
4.2: queryRec (s, sr)
r :recommender
4.3: repRec (s, sr, null, null)
Case (d): Unknown and recommendation not available

Fig. 4 Interaction diagrams elaborating different cases of the trust-based service collaboration algorithm

the server to share resources. Servers can employ certain trust rules for file storage such as `FileExcess` (uploaded files should have some acceptable size), `FileHarmful` (requestors should not upload any file containing malicious scripts), `FileSpam` (requestors should not upload insignificant files to the server to waste server space). Due to the gravity of the corresponding outcome, `FileHarmful` can be given high importance and `FileExcess` and `FileSpam` can be considered of medium and low importance respectively. File servers exchange recommendations, use CAT, and employ the algorithm to decide about service requests. A client is a service requestor (sr), the requested service (s) is `UploadDocFile`, and the file sharing server is the service provider (sp). We assume that sp uses 0.52 as the interaction threshold ($I.T.$). The actual value of $I.T.$ will depend on the particular target system. In the next subsections, we discuss the following four cases that will occur in the presented algorithm (see Fig. 4): (a) sr has previously uploaded `Doc` files to sp; (b) sr has not uploaded docs to sp, but has uploaded other files (e.g., `PDF`) previously; (c) sr has not interacted with sp at all previously, but has uploaded files to other file servers in the gird; (d) sr has not previously interacted with any server in the grid.

5.2.1 Case (a): Known for the same service

This case is elaborated in Fig. 4a. Upon the request for service s from sr, sp finds that sr has previously interacted with sp (1, 1.1). sr has uploaded `Doc` files to sp (1.2). sp then makes queries for recommendations and gets a reply from a recommender (r) (1.3, 1.4), where 0.8 is the recommendation value from r, and 2 is the recommendation path-length. Therefore, this a direct recommendation. The direct trust of sp on sr for s is 0.8 after applying time-based ageing on the direct trust retrieved from the repository (1.5). The recommendation accuracy of r to sp is 1 for s and sr (1.6). sp uses its self-confidence $\alpha = 0.8$, and calculates total trust T, which is 0.8 (1.7). Since this value is greater than $I.T.$, the service request is accepted (1.8).

5.2.2 Case (b): Known but not for the same service

This case is presented in Fig. 4b and is similar to Case (a), except that sp has no previous interaction record for s with sr (2.1, 2.2). However, sp finds that sr has uploaded `PDF` files previously. Applying Eq. 5, sp obtains a similar direct trust value for s and applies time-based ageing on it (2.3). sp obtains a direct recommendation 0.7 from r with recommendation accuracy 0.9 (2.4, 2.5, 2.6). sp uses $\alpha = 0.8$ and calculates T using Eq. 17 getting a value of 0.438 which is less than $I.T.$; so, the request is rejected (2.7, 2.8).

5.2.3 Case (c): Unknown but recommendations available

Figure 4c elaborates this case, where sr is unknown to sp (3.1), but sp finds an indirect recommendation of 0.9 from r, where the visited path-length is 3 with accuracy 0.96 for s (3.2, 3.3, 3.4, 3.5). sp uses the maximum allowed recommendation path $\Lambda = 5$, with $\Psi = 0.2$. Since sr is unknown to sp, it uses a low self-confidence value, $\alpha = 0.3$ in its total trust

Table 1 Elicited trust scenarios for a file sharing server

Trust scenarios	Description
File excess	Requestors may upload files beyond the limit of the server to make the upload service unavailable for others
File spamming	Requestors may upload illegal files to waste server space
File harmful	Requestors may upload files containing malicious scripts
Illegal access attempt	Requestors may try to access others' personal files in the resource database by manipulating the file search service
Remote file inclusion	Requestors may manipulate the file open service to open malicious files remotely and to execute them on the server

calculation. The total trust of sr thus becomes 0.58 which is slightly greater than $I.T.$; so, the request is granted (3.6, 3.7).

5.2.4 *Case (d): Unknown and recommendations not available*

This case is presented in Fig. 4d, and is almost similar to Case (c), but sp does not receive any recommendation for sr about s (4.4). Therefore, sp concludes that sr is a completely new client. sp assigns a neutral value of 0.5, does not examine $I.T.$, and grants the service request (4.5, 4.6).

6 Implementation and experimental evaluation

To confirm the applicability of the proposed service collaboration approach, we have developed a prototype file sharing grid, having each provider software incorporated with the trust model following the algorithm. We have generated different scenarios and tested the system against the scenarios.

6.1 Prototype implementation

In the prototype service-based grid system, there are a number of file sharing servers and clients, having each file sharing server incorporating the trust model and the algorithm. We focused on three types of file sharing service: file upload, open and search. Since file download takes place in the requestor side, we did not consider trust scenarios related to file downloading. The system runs on a Windows XP, Pentium 1.886 GHz, 1 GB RAM. The development language is Java, with the services described and configured in XML. The development platform is Jade 3.5 [27] on top of Eclipse IDE 3.2 [28].

The `Trust Rules` repository is developed by creating trust rules from different trust scenarios [19] of file sharing applications, listed in Table 1, where the left column provides the name of the trust scenarios and the right column describes them. These trust scenarios are some of the most common concerns in file sharing applications [19]. The `Direct Trust`, `Recommendations`, and `Recom-Accuracy` repositories are implemented as tables in a MySQL 5.0 database [29]. The `Direct Trust` repository has four fields $\{sr, s, T_D, t\}$, where sr is the service requestor which has interacted with the service provider for service s at time t. Based on the interaction analysis, T_D provides the trustworthiness of sr for s at time t. For example, $\{sr_1, UploadDocFile, 0.78, 1000\}$ implies that the sr_1 client interacted with the service provider based on the service $UploadDocFile$ at system time 1000, and the direct trust value after the interaction is 0.78. The `Recommendations` table has six fields $\{recID, sr, s, rV, M, t\}$, where $recID$ is the ID of the recommender which provided a recommendation on service requestor sr for service s at time t. The recommendation value is rV and the maximum visited path length is M. If M is greater than 2, then the recommendation is indirect. The `Recom-Accuray` table has five fields $\{recID, sr, s, A, t\}$, where $recID$ is the recommender ID with recommendation accuracy A at time t on service requestor sr for service s. Apart from these repositories, each service provider has another repository called `Similar` for trust calculations. This table has three fields $\{s, s_{similar}, sVal\}$, where s is the service which is similar to the service $s_{simialr}$, and the similarity measurement is denoted by $sVal$. The similarity calculation are performed based on the Eq. 7. Since the context-similarity parameter is reflexive, the first two entries in the `Similar` table (i.e., s and $s_{similar}$) can be used interchangeably.

6.2 Evaluating the service collaboration approach

We have incorporated each the calculation schemes of CAT in each of the file sharing servers of the prototype. The values of different constants are configured using XML file and changed dynamically at run-time. The default constant values used in our implemented system are as follows. The value of w_b in calculating confidence is 0.8 to emphasize that the users should be provided with more chances even if they were untrustworthy before. The value of δ in calculating direct trust is 0.8 to give more weight to previous confidence values. This means that a single recent untrustworthy interaction will not outweigh a history of trustworthy interactions and the entity may be given another chance. The value of Υ in

Table 2 Values of different constants used in experiments

Constant value	Constant value	Constant value	Constant value
$w_b = 0.8$ (Eq. 3)	$\delta = 0.8$ (Eq. 4)	$\Upsilon = 0.2$ (Eq. 8)	$\eta = 1$ (Eq. 9)
$\Lambda = 10$ (Eq. 11)	$\Psi = 0.2$ (Eq. 11)	$\zeta = 0.8$ (Eq. 12)	$\alpha = 0.8$ (Eq. 17)

time-based ageing parameter is 0.2 because in our system, there were only a few interactions (e.g., 100 at a time). η is chosen as 1 to provide a direct recommendation with more accuracy. Λ is 10 in path-based ageing parameter, since we have created in total 10 service providers. Ψ is 0.2 in path-based ageing parameter, since all the service providers in our system are considered almost trustworthy. The value of ζ in updating recommendation accuracy is 0.8 to give more weight to previous recommendation accuracy. It is chosen since a recommender can be wrong recently, however that cannot be used to decide its reliability. The value of α in the total trust calculation is 0.8 to emphasize more on self-observation and is used as 0.3 when service providers have had no direct interaction with service requestors. The constant values are summarized in Table 2, where each column provides the constant values used in experiments along with their corresponding equations. Although the constant values used in the experimental evaluation are chosen after careful considerations, they might change depending on the target service-based system. We evaluated the service collaboration algorithm based on its four different cases (recall Sect. 5.2), as explained below.

Case (a): Known for the same service. In the first case, the requestor sr_1 is known and has interacted previously for the requested service `UploadDocFile`. To test this case, we store a direct trust value of sr_1 in the `Direct Trust` repository for the service. When the request is received, the `Direct Trust` repository is searched for possible values. Since a direct trust value is already there, the client is considered known for this service. The provider sp_1 then queries for new recommendations (rRQ) to other providers (sp_2, sp_3, sp_4) and receives rRP accordingly. sp_4 does not have the requested value, and so it requests sp_5. Based on the reply from sp_5, sp_4 forwards it to sp_1. Since this recommendation from sp_5 has the visited path-length of 3, it is considered as an indirect recommendation. The recommendations of sp_2, sp_3 and sp_5 are stored in the `Recommendations` repository, and their corresponding previous recommendation accuracies are taken in `Recom-Accuracy`. If there is no previous recommendation accuracy available for a particular recommender on a requestor about a requested service, it is considered as a completely new recommender on sr_1 for $UploadDocFile$. The recommendation accuracy for this new recommender is set to 1 by default (i.e., initially, the recommender is considered as the most accurate on sr_1 for $UploadDocFile$). The total trust is calculated and the compared against interaction threshold ($I.T.$) of this service. The result is used to construct and send the service reply to sr_1, such as $\{sr_1, UploadDocFile, Accept, InteractionTime\}$.

Case (b): Known but not for the same service. The service requestor is sr_3 and the service s is `UploadDocFile`. However, the sr_3 has previous interaction record with the sp_1 for `UploadPDFFile` service. The sp_1 has the repository `Similar`, where the similarity between the two services are listed as 0.50 (If there are a number of similar services, we consider the service with the highest similarity value). The program execution flow then follows exactly as in the first case.

Case (c): Unknown but recommendations available. In this third case, the service requestor is sr_5 and the requested service s is `UploadDocFile`. sr_5 has had no previous interaction with sp_1 based on the `Direct Trust` repository, i.e., sp_1 has no entry in the `Direct Trust` repository for sr_5. New recommendations are queried and found. Since the requestor is unknown to sp_1, any recommendation regarding the sr_5 is considered to have accuracy 1 by default. However, the value of α is reduced to 0.3 in this case since sp_1 has no previous interaction value for sr_5.

Case (d): Unknown and recommendations not available. In the fourth case, the service requestor is sr_6 who is completely new to sp_1 and to all recommenders. Therefore, the Trust Engine provides a default direct trust value 0.5 to sr_6 for the requested service s and notifies the requestor accordingly. The direct trust value is stored in the `Direct Trust` repository. After this case is executed, the requestor becomes known to the system. Therefore, next time the user requests the same service, it is treated as one of first two cases [i.e., Cases (a) or (b)].

6.3 Performance overhead

To investigate the performance overhead in using the trust model, we conduct three experiments. The first experiment concludes that there is some delay in providing trust-based service granting decision on service request events, although the average delay remains almost constant with the increase in the number of service requests. The second experiment shows that the run-time monitoring and analysis of service-based interactions do not create that much overhead. It should be noted that the purpose of these experiments is to show that the performance overheads remain almost static, which makes the approach applicable for large-scale systems. Therefore, the focus of the first two experiments was not

Table 3 Property-based placing of different trust-based collaborative approaches

Model	RO	CA	RA	RB	TA	PA	C/D	RF	CO	VB	SB
FIRE [8]	Y	Y	Y	Y	N	N	D	N	N	Y	Y
FTMAS [9]	N	N	N	Y	N	N	D	N	N	Y	N
TDAS [15]	N	Y	N	Y	N	N	D	Y	N	N	N
TSOC [10]	N	Y	N	Y	N	N	D	N	Y	N	Y
CertainTrust [11]	N	Y	Y	Y	N	N	D	N	N	Y	Y
FuzzyWeb [12]	N	Y	Y	Y	N	N	D	Y	Y	N	Y
BambooTrust [31]	N	Y	Y	N	N	N	C	N	N	N	N
SPORAS [30]	N	N	N	Y	N	N	C	N	Y	Y	Y
REGRET [14]	N	Y	N	Y	N	N	D	N	N	Y	Y
ICRM [36]	N	N	N	Y	N	N	D	Y	N	Y	Y
TRAVOS [35]	N	N	N	Y	N	N	D	Y	N	Y	Y
TRP2P [22]	N	Y	N	Y	N	N	D	Y	N	N	Y
TBRM [23]	N	Y	Y	Y	N	N	D	N	N	Y	Y
TRBAC [2]	Y	Y	Y	Y	N	N	D	N	N	N	N
TBAC [7]	Y	Y	Y	Y	N	N	D	Y	N	N	N
TrustBAC [39]	Y	Y	Y	Y	N	N	D	N	N	N	N
ATP [13]	N	N	N	N	N	N	D	N	N	Y	Y
OFTM [5]	N	Y	Y	Y	N	Y	D	N	N	N	Y
EMDR [38]	N	N	N	Y	N	N	D	Y	N	Y	Y
PeerTrust [40]	N	Y	N	Y	N	N	D	Y	N	Y	Y
Beta [32]	N	Y	N	Y	N	N	C	N	N	N	Y
B-trust [16]	N	Y	Y	Y	N	N	D	N	N	N	Y
CAFDS [26]	N	Y	Y	Y	N	N	D	N	Y	N	Y
TMP2P [25]	N	Y	N	Y	Y	N	D	Y	N	Y	Y
ICTM [33]	Y	Y	N	N	N	N	C	N	N	N	N
P2PTC [41]	Y	N	N	N	N	N	D	N	N	N	N
ATN [34]	Y	N	N	N	N	N	C	N	N	N	N
EigenTrust [42]	N	Y	N	Y	N	N	D	Y	N	N	N
SureMsg [43]	N	N	Y	Y	N	Y	D	N	N	N	N
FuzzyTrust [37]	Y	Y	Y	Y	N	N	D	N	N	N	N
SPM [44]	N	N	Y	N	N	N	D	N	N	N	Y
TAP [6]	Y	N	Y	N	N	N	D	N	N	Y	N
Our Work	Y	Y	Y	Y	Y	Y	D	Y	Y	Y	Y

The trust properties follow the definitions provided in Sect. 3.
RO rule-oriented, *CA* context-aware, *RA* risk-aware, *RB* recommendation-based, *TA* time-based ageing, *PA* path-based ageing, *C/D* centralized/decentralized, *RF* recommendation-filtration, *CO* closeness-oriented, *VB* variance-based, *SB* sample-based

to show the difference between the two response time delays that occur in the system with and without the monitor. The third experiment indicates that a large recommendation chain is probably not a good idea when there is a need for prompt reply to service request events.

7 Related trust-based collaborative approaches

Trust-based collaboration has been addressed to analyze the trustworthiness of potential stakeholders and make trust-based decision. Table 3 compares and contrasts the related collaborative approaches; the first column lists the work and the rest of the columns analyze the work based on the collaboration-based trust properties of Sect. 3. The three properties, semi-transitiveness, non-symmetricness, and dynamism (see properties P6, P7 and P8, respectively), are common-sense theoretical properties of trust relation that are likely provided by most of all of the work. Therefore, they are not included in this table. The table indicates that our approach satisfies all of the ten desirable properties, while no other approach satisfies more than six. Besides our model, only OFTM [5] satisfies path-based ageing, and our model is unique in its implementation of time-based ageing. Moreover, no other work propose any trust-based service collaboration algorithm. Each of the related work has been briefly discussed in the following paragraphs.

In the eBay trust and reputation model [24], users can provide feedback ratings after each interaction. The model is not context-aware, so big and small transactions are all treated equally. SPORAS [30] proposes a solution to this problem by calculating and assigning trust based on rules. However, it discourages newcomers by providing very low initial values. In the BambooTrust [31] reputation system, users can query about another user. The query requests are handled in parallel machines to provide fast reply. Beta model [32] uses statistical representation of trust and reputation for e-commerce systems based on beta probability density functions. A trusted third party is introduced in [33] to monitor between service providers and requestors from trust perspectives, while [34] proposes negotiation between service providers and requestors based on the corresponding credentials. FIRE [8] selects providers based on previous interaction records and recommendations. Although FIRE uses interactions to gather trust, it does not rely on rules specifically to monitor it. Billhardt et al. [10] propose a trust model for selecting a provider in service-oriented computing. Toivonen et al. [26] propose an ontology-based context similarity method. Similar to them, we propose a context-similarity parameter, however our parameter uses word-based similarity calculation while they use organization-level service hierarchy.

TRAVOS [35] is a trust and reputation mechanism based on direct interactions and recommendations, adding the ability to identify malicious recommendations. However, they neither apply rules to analyze interactions, nor use time-based and path-based parameters. A broker based decentralized reputation system is proposed in [36], which unlike us penalizes entities for giving false recommendations. Certain trust [11] separates humans from software entities and provides mapping between them for dynamic decision making. Fuzzy notations are used by [12,37] to reason about trust or to propagate trust information while choosing an entity. Bayesian trust models are used by [22] and B-trust [16] to denote the context-awareness nature of trust. Combining ideas from each, we use context-aware, rule-based trust reasoning in CAT.

In REGRET [14], users can rate each other after each interaction. It proposes a time-based ageing parameter. However, the parameter takes only the ratio of current and previous times and does not consider the system environment. An automatic trust prediction model is used in [13] to select providers, based on their trustworthiness from the advertised and monitored attributes. Trust and recommendations are formally defined in [9,15,38] by incorporating belief, disbelief and uncertainty. Honesty, recommendations and recommendation accuracies are used in [25] to provide access to peer-to-peer system resource.

Trust-based decision making for a service request on the server side is mostly used to provide access to system resources. Dimmock et al. [7] propose trust-based access control (TBAC) for by defining policies. The server uses trust and recommendation values in its authorization policies to infer the trustworthiness of the clients before providing them services. Similarly, Chakraborty and Ray [39] propose TrustBAC, a trust-based software access control model, and Sandhu and Zhang [41] present a hardware control for trusted computing. These models provides the detailed structure of permission messages, including user name, trust values in the assigned role. While all of of these work look at the access control mechanism from authorization perspectives, we focus on service granting. Unlike our work, neither of the above propose any algorithm. TRBAC [2] proposes access control policies to server resources based on trust reasoning. While both of the TRBAC [2] and TrustBAC [39] are risk-aware, unlike our system, they do not use time-based ageing, service similarity, and recommendation filtration.

Lin and Varadharajan [23] integrate trust with risk analysis to grant authorization. Haque and Ahamed [5] propose OFTM, a formal trust model for pervasive computing to handle service requests in mutual collaboration. We adapt path-based ageing from [5] by proposing a different calculation scheme. PeerTrust [40] is a context and risk-aware trust model that operates on the online rating mechanisms for e-commerce transactions, such as on eBay; unfortunately it does not offer any of the other desirable trust properties.

Rajbhandari et al. [6] use XML rules to measure the trustworthiness of service providers, while English et al. [44] propose a monitoring architecture to use with a service providing software. Unlike us, they do not propose any trust model for quantifying collaboration. A reputation-based algorithm is proposed by [42] to propagate trust information over a network, assuming that the trust values are presently available. Emails from untrustworthy users are discarded by [43] based on the feedback from all the users in the system. Unlike the algorithms provided by [42,43], we accept or reject service requests from trust perspectives.

Although all these approaches mention trust models for quantifying trustworthiness in certain system environments, they actually do not guide the development of trust relationship specifications in the context of trust rules. For example, they do not consider whether a system meets user trust requirements or whether a system uses trust in its own development phase. Yu and Liu [45] address these issues at the requirement level of system development by using trust as a non-functional requirement, where trust is a combination of all or some quality attributes of a system. They illustrate their approach by describing the behavior of a system in the presence of attack, and examine required defenses from trust perspectives. Horkoff et al. [46] extend their work in the trusted computing base, by claiming that trust can be included in the technological framework of a system at both the software and the hardware level. We follow the approach of [45, 46] in [19,47] by capturing security as a system trust requirement and specifying trust rules based on the prevalent trust concerns in the system; we consider an entity as secure if it is safe under malicious activities or attacks. In [19], we propose a Unified Modeling Langauge (UML) extension, called UMLtrust (UML for trust scenarios) to identify system trust scenarios and specify the scenarios in trust rules. In [47], we propose a software development framework to incorporate UMLtrust in trust-aware service-based software development environment. The evaluation of [19,47] is performed using different trust scenarios from service-based software systems.

8 Conclusions and future work

Given the fact that service providers are exposed to users from multiple organizational domains and so can be exploited by untrustworthy users, the service collaboration in such open and dynamic systems needs to incorporate trust in service-based interaction analysis and decision making. To address this, we propose a trust-based service collaboration approach by quantifying service collaboration using our CAT (Context-Aware Trust) model and by facilitating automatic trust-based service granting using a service collaboration algorithm. We identify some common collaboration-based trust properties and integrate those to CAT. We introduce a context-similarity and time-based ageing parameters in direct trust calculation. We propose direct and indirect recommendations, applying a path-based ageing parameter on indirect recommendations. A mechanism to calculate the accuracy of recommendations is described. The service collaboration algorithm uses CAT to make service granting decision on behalf of a service provider entity. The approach is explained, implemented and evaluated using examples from file sharing grid.

The approach presented in this paper can be used with traditional access control mechanisms in a complementary fashion. Most access control mechanisms focusing mainly on authorization aspects can be enhanced based on the time-based ageing, service similarity, and recommendation filtration techniques. In other words, the existing authentication and authorization can be followed by quantified service collaboration from trust perspectives to decide on service requests at run-time based on our trust-based service collaboration algorithm.

Our future enhancements to the system will concentrate on addressing the following limitations of our research work. In our work, it is considered that the network running a trust model is secure from false recommenders, and therefore, the recommendations from an entity are considered truly from that entity. However, in real situations, this might not be the case. For example, we do not consider defense against the Sybil attacks [48], where a malicious entity creates a large number of artificial entities for the purpose of providing false recommendations to influence trust-based decision making. We consider that this problem of pseudonymity can be handled by an authentication system that is responsible for providing unique identifications to the interacting entities. For example, in a resource sharing grid, the users belong to particular organizations [7], with the organizations subscribing for particular services. We use CAT for evidence-based trust decision making, where the trust rules are used to monitor the interactions between the entities. Based on the monitoring, the trustworthiness of the interacting entities is measured. However, CAT measures the accuracies of the corresponding recommendations after each interaction. Since the measurement of accuracy is performed by calculating the difference between the direct interaction trust and the recommendations, unreliable recommendations are detected and discarded accordingly. In this way, we can also partially address the colluding malicious node attack [49], where malicious nodes collectively provide false recommendations about an entity.

We also plan to incorporate CAT into pervasive computing applications for dynamic service provision. In this respect, we will specifically focus on the efficiency of our algorithm based on its performance overhead in different execution scenarios of pervasive environments.

References

1. Durad M, Cao Y (2006) A vision for trust managment grid. In: Proceedings of the sixth IEEE international symposium on cluster computing and the grid, Singapore. IEEE CS Press, Los Alamitos, pp 34–44
2. Dimmock N, Bacon J, Ingram D, Moody K (2005) Risk models for trust-based access control. In: Proceedings of the third annual conference on trust management, Lecture Notes in Computer Science, vol 3477. Springer, Heidelberg, pp 364–371
3. Gambetta D (1988) Can we trust trust? In: Gambetta D (ed) Trust: making and breaking cooperative relations, Chap 13. University of Oxford, 213–237
4. Bond A, Arnold D (1994) Visualizing service interaction in an open distributed system. In: Proceedings of the first international workshop on services in distributed and networked environments, Prague, Czech Republic. IEEE CS Press, Los Alamitos, pp 19–25
5. Haque M, Ahamed SI (2007) An omnipresent formal trust model (FTM) for pervasive computing environment. In: Proceedings of the 31st annual IEEE international computer software and applications conference, Beijing, China. IEEE CS Press, Los Alamitos, 49–56
6. Rajbhandari S, Contes A, Rana OF, Deora V, Wootten I (2006) Trust assessment using provenance in service oriented applications. In:

Proceedings of the tenth IEEE on international enterprise distributed object computing conference workshops, Hongkong. IEEE CS Press, Los Alamitos, pp 65–72
7. Dimmock N, Belokosztolszki A, Eyers D, Bacon J, Ingram D, Moody K (2004) Using trust and risk in role-based access control policies. In: Proceedings of the ninth ACM symposium on access control models and technologies. ACM Press, New York, pp 156–162
8. Huynh T, Jennings N, Shadbolt N (2004) FIRE: An integrated trust and reputation model for open multi-agent systems. In: Proceedings of the 16th European conference on artificial intelligence. Valencia, Spain, pp 18–22
9. Wang Y, Singh M (2007) Formal trust model for multiagent systems. In: Proceedings of the 20th international joint conference on artificial intelligence. India, pp 1551–1556
10. Billhardt H, Hermoso R, Ossowski S, Centeno R (2007) Trust based service provider selection in open environments. In: Proceedings of the 22nd annual ACM symposium on applied computing, Seoul, Korea. ACM Press, New York, pp 1375–1380
11. Ries S (2007) Certain trust: a trust model for users and agents. In: Proceedings of the 22nd annual ACM symposium on applied computing, Seoul, Korea. ACM Press, New York, pp 1599–1604
12. Sherchan W, Loke S, Krishnaswamy S (2006) A fuzzy model for reasoning about reputation in web services. In: Proceedings of the 21st annual ACM symposium on applied computing, Dijon, France. ACM Press, New York, pp 1886–1892
13. Capra L, Musolesi M (2006) Autonomic trust prediction for pervasive systems. In: Proceedings of the 20th international conference on Advanced Information Networking and Applications. Vienna, Austria, IEEE CS Press, Los Alamitos, 481–488
14. Sabater J, Sierra C (2001) REGRET: A reputation model for gregarious societies. In: Proceedings of the fourth workshop on deception, fraud and trust in agent societies, Canada, pp 61–70
15. Wang Y, Singh M (2006) Trust representation and aggregation in distributed agent systems. In: Proceedings of the 21st international conference on artificial intelligence, Boston
16. Quercia D, Hailes S, Capra L (2006) B-trust: Bayesian trust framework for pervasive computing. In: Proceedings of the fourth international conference on Trust Management, Lecture Notes in Computer Science, vol 3986, Springer, Heidelberg, pp 298–312
17. Jøsang A (1996) The right type of trust for distributed systems. In: Proceedings of the 1996 workshop on new security paradigms, Lake Arrowhead. ACM Press, New York, pp 119–131
18. Uddin MG, Zulkernine M, Ahamed SI (2008) CAT: a context-aware trust model for open and dynamic systems. In: Proceedings of the 23rd annual ACM symposium on applied computing, Brazil. ACM Press, New York, pp 2024–2029
19. Uddin MG, Zulkernine M (2008) UMLtrust: towards developing trust-aware software. In: Proceedings of the 23rd annual ACM symposium on applied computing, Brazil. ACM Press, New York, pp 831–836
20. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J High Perform Comput Appl 15(3): 200–222
21. Tie-Yan L, HuaFei Z, Kwok-Yan L (2003) A novel two-level trust model for grid. In: Information and communications security, Lecture Notes in Computer Science, vol 2836. Springer, Heidelberg, pp 214–225
22. Wang Y, Vassileva J (2003) Trust and reputation model in peer-to-peer networks. In: Proceedings of the third international conference on peer-to-peer computing, Sweden. IEEE CS Press, Los Alamitos, pp 150–157
23. Lin C, Varadharajan V (2006) Trust based risk management for distributed system security—a new approach. In: Proceedings of the first international conference on availability, reliability and security. Vienna, Austria, IEEE CS Press, Los Alamitos, pp 6–13
24. Grandison T, Sloman M (2000) A survey of trust in internet application. in IEEE Communications Surveys & Tutorials 3(4):September 2000. 15pp
25. Azzedin F, Maheswaran M (2003) Trust modeling for peer-to-peer based computing systems. In: Proceedings of the international symposium on parallel and distributed processing. IEEE Press, New York, 10pp
26. Toivonen S, Lenzini G, Uusitalo I (2006) Context-aware trust evaluation functions for dynamic reconfigurable systems. In: Proceedings of models of trust for the Web. Edinburgh, Scotland
27. Bellifemine F, Caire G, Poggi A, Rimassa G (2003) Jade: a white paper. Exp Search Innov 3(3):14
28. Eclipse (2008) Eclipse 3.2 Documentation
29. MySQL Enterprise Server (2008) MySQL 5.0 Reference Manual
30. Zacharia G, Maes P (2000) Trust management through reputation mechanisms. in Applied Artificial Intelligence 14(9): 881–908
31. Kotsovinos E, Williams A (2006) BambooTrust: practical scalable trust management for global public computing. In Proceedings of the 21st annual ACM symposium on applied computing, Dijon, France. ACM Press, New York, pp 1893–1897
32. Jøsang A, Ismail R (2002) The beta reputation system. In: Proceedings of the 15th bled conference on electronic commerce, Slovenia
33. Etalle S, Winsborough W (2005) Integrity constraints in trust management. In: Proceedings of tenth ACM symposium on access control models and technologies, Sweden. ACM Press, New York, pp 1–10
34. Ryutov T, Zhou L, Neuman C, Foukia N, Leithead T, Seamons K (2005) Adaptive trust negotiation and access control for grids, In: Proceedings of the sixth IEEE/ACM international workshop on grid computing, Washington, USA. IEEE CS Press, Los Alamitos, pp 55–62
35. Teacy W, Patel J, Jennings N, Luck M (2002) Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In: Proceedings of fourth international joint conference on autonomous agents and multiagent systems, pp 997–1004
36. Jurca R, Faltings B (2002) Towards incentive-compatible reputation management. In: Trust, reputation and security: theories and practice. Lecture Notes in Computer Science, vol 2631. Springer, Heidelberg, pp 138–147
37. Lesani M, Bagheri S. Applying and inferring fuzzy trust in semantic web social networks. In: Proceedings of the Canadian semantic web (semantic web and beyond computing for human experinece V2). Springer, Heidelberg, pp 23–43
38. Yu B, Singh MP (2002) An evidential model of distributed reputation mechanism. In: Proceedings of first international conference on autonomous agents and multiagent systems. ACM Press, New York, pp 294–301
39. Chakraborty S, Ray I (2006) TrustBAC: integrating trust relationships into the RBAC model for access control in open systems. In: Proceedings of the 11th ACM symposium on access control models and technologies. California, USA. ACM Press, New York, pp 49–58
40. Xiong L, Liu L (2002) Building trust in decentralized peer-to-peer electronic communities. In: Proceedings of the fifth international conference on electronic commerce research. Montreal
41. Sandhu R, Zhang X (2005) Peer-to-peer access control architecture using trusted computing technology. In: Proceedings of the tenth ACM symposium on access control models and technologies, Sweden. ACM Press, New York, pp 147–158
42. Kamvar SD, Schlosser MT, Molina-Garcia H (2003) The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th international conference on world wide web, Budapest. ACM Press, New York, pp 640–651
43. Zhang W, Bi J, Wu J, Qin Z (2007) An approach to optimize local trust algorithm for suremsg service. In: Proceedings of the 2007 ECSIS symposium on bio-inspired, learning, and intelligent

systems for security, Edinburg, IEEE CS Press, Los Alamitos, pp 51–54
44. English C, Terzis S, Nixon P (2005) Towards self-protecting ubiquitous systems: monitoring trust-based interactions. Pers Ubiquitous Comput 10(1):50–54
45. Yu E, Liu L (2001) Modelling trust for system design using the i* strategic actors framework. In: Proceedings of the international workshop on deception, fraud, and trust in agent societies, Lecture Notes in Computer Science, vol 2246. Springer, Heidelberg, pp 175–194
46. Horkoff J, Yu E, Liu L (2006) Analyzing trust in technology strategies. In: Proceedings of the international conference on privacy, security, and trust. McGraw-Hill, New York, pp 21–32
47. Uddin MG (2008) Development and automatic monitoring of trust-aware service-based software systems. In: Queen's University Masters Thesis
48. Douceur J (2002) The sybil attack. In: Proceedings of the first international workshop on peer-to-peer systems, Lecture Notes in Computer Science, vol 2429. Cambridge, MA, USA, Springer, Heidelberg, pp 251–260
49. Damon M, Doug S, Dirk G (2007) A mechanism for detecting and responding to misbehaving nodes in wireless networks. In: Proceedings of the fourth IEEE communications society conference on sensor, mesh and ad hoc communications and networks. IEEE CS Press, Los Alamitos, pp 678–684