A Trust Model for the SoSIReCr Portal

Tomáš Knap

Department of Software Engineering
Charles University in Prague, Czech Republic
tomas.knap@mff.cuni.cz

Abstract. This document has two main Parts - Part 1 describes the Trust model and its application in the SoSIReCr portal; Part 2 sketches the architecture and implementation of the Trust subsystem for the SoSIReCr portal implementing the Trust model proposed in Part 1.

1 Organization of the Document

Part 1 – The Trust model and its application in the SoSIReCr portal (Sections 2 – 7):

- Section 2: Motivation for trust, concept of trust
- Section 3: Social trust network - a graph model representing entities and trust relations between them in the SoSIReCr portal
- Section 4: Defining a trust algorithm GriTa for deriving trust between arbitrary two entities in the social trust network
- Section 5: Extension of the trust algorithm GriTa with reputations of entities
- Section 6: Application of the Trust model (introduced in Sections 2 – 5) to the activities in the SoSIReCr portal according to the requirement analysis
- Section 7: Related Work in the field of trust models and algorithms

Part 2 – The sketch of the architecture and implementation of the Trust subsystem for the SoSIReCr portal (Sections 8 – 9):

- Section 8: The architecture of the Trust subsystem
- Section 9: Implementation Notes

2 Concept of Trust

Since anybody can join the SoSIReCr portal, a mechanism which will help entities to decide the trustworthiness of the profiles and professional profiles of other entities is highly important – it helps the entities to deduce the suitability of the other entities as team/project participants, employees, or as entities whose profiles and professional profiles are worth taking into account.

Manifestations of trust are easy to recognize because we experience and rely on it every day, but at the same time trust is quite challenging to define because it manifests itself in many different forms and it is used with wide variety of meanings [10, 1, 6, 16]. Definition 1 drives the comprehension of trust in the Trust model detailed in further text.

Definition 1. ***Concept of (general) trust***. *Suppose that an entity* ***u*** *encounters a situation where it perceives an ambiguous path (has to decide whether to use the data from the profile or professional profile of the entity* ***v****). The result of following the path can be good or bad, and the occurrence of the good or bad result is contingent on the data in the profiles of the entity* ***v****. If the entity* ***u*** *chooses to go down the path (decides to use data in the profiles of the entity* ***v****), it has made* ***a trusting choice****; it trusts that the entity* ***v*** *has filled its profile and professional profile according to its best conscience – the data in the profiles is impartial, corresponds with the reality, and the data is not misleading.*

Many publications (such as [9, 5, 7]) emphasize that the general trust (as introduced in Definition 1) is not sufficient, because trust is context dependent. We agree with this objection – suppose you are a project manager searching an entity with a particular expertise to fill up the project team; in that case, general trust is beneficial, however, not sufficient. Two solutions to this problem are possible.

The first approach defines the concept of *context trust* of an entity in another entity w.r.t. the particular context – axis of the professional profile [12] – and enable the entities to express context trust in the SoSIReCr portal. In this approach, it is hardly imaginable that the entities will explicitly specify context trust in a substantial amount of entities and axes of the professional profiles. Furthermore, every change to an axis of the entity's professional profile should alarm other entities having a context trust to that axis, because that change can influence the amount of context trust. Furthermore, if we redesign the axes in professional profiles, we should notify all entities that new axes are available and old axes are no more used – this redesigning process reduces the number of trust expressions and, thus, suppresses the effect of the Trust model and, what is more important, it demotivates entities to express any context trust.

Alternatively, we enable the entities to express a (general) trust of an entity u in another entity v and combine the general trust with the results of further context-based analyses, such as the analysis of experience and expertness of the entity v w.r.t. the particular axis of the professional profile (the experience and expertness of the entity can be based on the value on the given axis and evidence conducted – published articles, implemented tools, finished projects). Based on the general trust enabling us to judge whether the professional profile (the input to these analyses) is trustworthy and to what extend and the result of these analyses, we can deduce the context trust of the entity u in the entity v. In this approach, the context trust is computed automatically, based on the data in professional profiles. We can easily add other analyses and factors influencing the final context trust, such as a location of the entity, the entity's track record (how successful was the entity's output in the similar tasks in the past) etc. Furthermore, the profiles and professional profiles of entities can evolve during the time; the only adjustment that must be done is the adjustment of the particular context-based analysis.

In the SoSIReCr portal, we have chosen the latter approach – the first version of the Trust model supports the general trust according to Definition 1, with a future extension to context trust in mind.

The *trusting choice* in Definition 1 can be quantified either on a discrete [14, 7] or continuous scale [15, 8, 20, 18, 7]. In general, discrete trust levels are easily seizable by humans when expressing trust between each other, on the other hand, continuous trust values provide more accurate expressions of trust, however, they are typically far below the level of differentiation of the average human.

In [7], Golbeck internally uses continuous trust values $tv \in [1, 10]$, however, externally, the entity is provided with ten discrete *trust levels* ranging from "absolute trust" ($tv = 10$) to "absolute distrust" ($tv = 1$), with $tv = 5$ expressing the "neutral trust" (neither positive, nor negative) or the absence of trust. We exploited the similar approach.

In our Trust model, the trusting choice of an entity u in the entity v is internally quantified as a continuous *trust (value)* $\tau_{u,v}^{(t_1,t_2)} \in [0, 1] \cup \bot$ which is valid in the given time interval (t_1, t_2) ($t_2 = \infty$ if the right bound is unknown). To simplify the formulas in further text, we define $\tau_{u,v}$ holding the current trust value as $\tau_{u,v} = \tau_{u,v}^{(t_1,t_2)}$, so that $t_2 = \infty \vee \neg\exists t'_2 > t_2$ ($\forall t \neq \infty : \infty > t$).

Table 1. Trust Levels

Trust Level Label	**Abbreviation**	$\boldsymbol{\tau_{u,v}}$
DISTRUSTED	D	0.1
KNOWN	K	0.3
TRUSTED	T	0.7
TRUSTED_HIGHLY	TH	0.9

The interval [0,1] for expressing the trust value $\tau_{u,v}$ is a a frequent representation of trust [18, 8, 20, 1, 6]. The trust value $\tau_{u,v}$ is ranging from "absolute distrust" ($\tau_{u,v} = 0$) to "absolute trust" ($\tau_{u,v} = 1$), the value $\tau_{u,v} > \kappa_{trusted}$ expresses that the entity v is considered as trusted (to some extend); $\tau_{u,v} = \bot$ if the trust value is unknown. The variable $\kappa_{trusted}$, by default $\kappa_{trusted} = 0.3$, depicts the adjustable threshold for rather trustworthy entities. We decided to let, by default, bigger part of the interval [0,1] for expressing trust values of rather trustworthy entities, because we propose two trust levels for rather trustworthy entities (see further).

We assume that $\tau_{u,u} = 1$. Although some researchers, like [15], do allow entities to assign the trust value representing 100% trust only to themselves, because an entity u cannot be 100% sure that another entity v will behave like expected; what is more, if $\tau_{u,v} = 1$, $u \neq v$, the trusting choice of the entity u in Definition 1 is missing – the entity u is blindly accepting the profile of the entity v, without any consideration. For simplification, we allow $\tau_{u,v} = 1$ – it is useful to express implicit trust relations between certain entities, such as group and its subgroup. The trust value $\tau_{u,v} = 0$ can be ascribed, however, typically, only through a thoughtful judgement based on the bad experience of the entity u with the entity v in the past [15] – the entity u has absolutely distrust in the profile of v.

We agree that discrete trust levels are easily seizable by the human entities, which form the majority of entities in the SoSIReCr portal. Regrettably, it is hard to imagine that human entities will consistently subjectively map their trust to others on a ten point scale (as proposes Golbeck in [7]) – the same entity perceived by several entities as "really trusted" is likely to receive different trust levels from these entities. Therefore, we externally use only four trust levels depicted in Table 1 together with the corresponding abbreviations and trust values $\tau_{u,v} \in [0,1]$. The level *TRUSTED_HIGHLY* should be given to close friends of v, family members, colleagues in a company, or the working group of the entity – the entity u is almost sure that these entities has filled their profiles according to their best conscience. The trust level *TRUSTED* should be assigned to all other entities not classified as TRUSTED_HIGHLY, however, still rather trustworthy – we can expect that they have filled their profiles according to their best conscience. If the entity u only knows another entity v, however, cannot decide whether v is rather trustworthy or untrustworthy, the trust level *KNOWN* should be used. If the entity u wants to explicitly express that the entity v is distrusted, trust level *DISTRUSTED* is used.

3 Social Trust Networks

Based on the definition of the concept of trust (Definition 1) and the trust levels proposed (Table 1), we formalize the concept of a social trust network – a graph model behind the SoSIReCr portal representing entities and trust relations between them.

A *social trust network*[1] is a directed weighted graph $stn_t(V, E, \tau_t) \in \mathcal{STN}$, where $V \subseteq \mathcal{V}$, $E \subseteq \{V \times V\}$, and $\tau_t : E \rightarrow [0,1]$ is the *trust function* defining the weights – trust values – for the edges; $tau_t((u,v)) = \tau_{u,v}^{(t_1,t_2)}$, $t \in (t_1, t_2)$, $u, v \in V$. If $t = NOW$, $t \in (t_1, \infty)$, we deal with the most current social trust network, abbreviated as $stn(V, E, \tau)$. Formally, a *trust relation* is an edge $e \in E$ in the social trust network $stn_t(V, E, \tau_t)$, together with its weight $\tau_t(e)$.

Example 1. *Suppose that Alice, a user of the SoSIReCr portal is part of the social trust network $stn(V, E, \tau)$ depicted in Figure 1. The social trust network consists of trust relations between Alice and (1) her colleague Bob working in the same group ($\tau((A,B)) = TH$), (2) other users, such as Cyril ($\tau((A,C)) = T$), and (3) the her working group ($\tau((A,WG)) = T$). Obviously, all the depicted entities can have trust relations to other entities not depicted in Figure 1.*

[1] A concept of social trust networks is similar to the web of trust introduced in PGP system

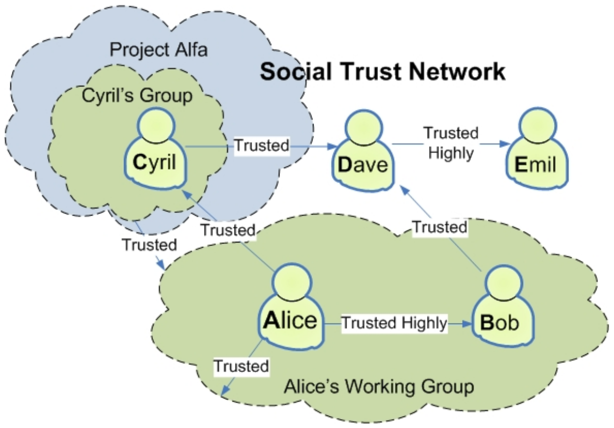

Fig. 1. Sample social trust network Alice is part of.

3.1 Explicit and Implicit Trust Relations

There are two types of trust relations in the social trust network $stn(V, E, \tau)$:

- Explicit: Entities can directly express their trust in other entities of the social trust network and the implementation must enable this.
- Implicit: Trust relations can be deduced automatically from the facts about the entity (data in profiles). Table 2 introduces for every type of considered fact stated by the entity u about the entity v the corresponding $\tau_{u,v}$ of the implicit trust relation.

The implicit trust relation is created when the particular fact from the table of types of facts (Table 2) is true. The implicit trust relations must not overwrite any already explicitly created trust relations. If the explicit trust relation (u, v) already existed between two entities, the implicit trust relation (u, v) is not created. The implicit trust relation cannot be manually deleted – it can be overridden by the explicit trust relation or it automatically disappears when the particular fact leading to its creation becomes invalid (e.g. the user is moved to another group, the project has finished). Figure 2 depicts the social trust network from Figure 1 with implicit trust relations added (for clarity, the weights on the edges are hidden); implicit trust relations are denoted with dashed lines.

The implementation must enable easy customization of the trust values $\tau_{u,v}$ in Table 2. In the future versions of the Trust model, the set of implicit trust relations can be extended (currently it includes users, groups, and projects).

4 Trust Algorithm GriTa

Suppose that entities u, v are not connected by a trust relation; then, the entity $u \in \mathcal{V}$ can either estimate only trustworthiness of its *neighbors* (entities connected by a trust relation in the social trust network $stn(V, E, \tau)$), or we need to

Table 2. Trust Levels

Types of Facts	**Trust Value** $\tau_{\mathbf{u},\mathbf{v}}$
A user u is a member of the group v	TH
A group u has a user v as the member	TH
A user u is a colleague of the user v in a project/group	TH
A group u participates in the project v	TH
A project u involves participating group v	TH
A group u cooperates with another group v on the same project	TH

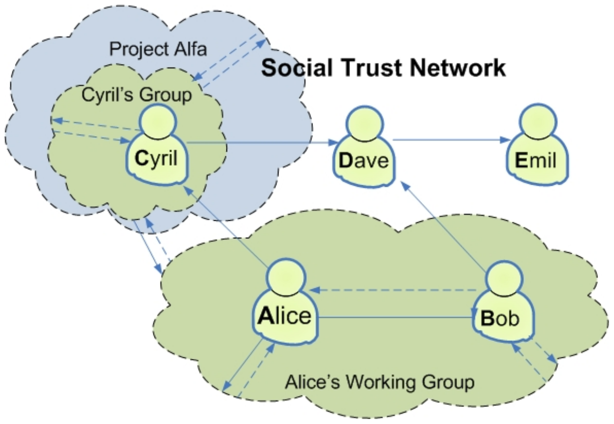

Fig. 2. Social trust network Alice is part of with the implicit trust relations added.

compute a quantitative estimate – *a mediated trust value* – of how much trust an entity u should accord to the entity v in *stn*. Before defining the trust algorithm GriTa used to compute the mediated trust value, we introduce trust relation paths.

The important question is whether trust should be transitive. As stated in [18]: *"[...] from the perspective of network security (where transitivity would, for example, imply accepting a key with no further verification based on trust) or formal logics (where transitivity would, for example, imply updating a belief store with incorrect, impossible, or inconsistent statements) it may make sense to assume that trust is not transitive"*. On the other hand, Golbeck [7] and Guha et al. [8] show experimentally that trust is transitive and may propagate along the trust relations in social networks. In contrary to Golbeck's approach, Guha et al. suggest appropriate trust discounting with the increasing lengths of trust relation paths between two entities, which is a reasonable property of trust – one trusts more his direct friends than friends of friends of his friends. Since the context and the networks in which trust is used by Guha and Golbeck are similar to the social trust networks proposed in Section 3, we comprehend trust as transitive with appropriate trust discounting.

Let us formally define an already mentioned concept – a *trust relation path*, $\pi_{u,v} \in \mathcal{TP}$, between entities u and v in some $stn(V, E, \tau)$ – as a progression of trust relations $(u = v_0, v_1), \ldots, (v_i, v_{i+1}), \ldots, (v_{n-1}, v_n = v)$, $v_i \in V$, $0 \leqq i \leqq n$; each $e_j = (v_j, v_{j+1}) \in E$ and $\forall e_j, e_k \in \pi_{u,v} : e_j \neq e_k$, where $0 \leqq j, k \leqq n-1$. The expression $e \in \pi_{u,v}$ ($x \in \pi_{u,v}$) denotes that $e \in E$ ($x \in V$) is part of the path $\pi_{u,v}$. The number of trust relations involved in $\pi_{u,v}$ is denoted as $|\pi_{u,v}|$, whereas $\#\pi_{u,v}$ ($\#\pi^{\chi}_{u,v}$) denotes the number of vertex-disjoint (edge-disjoint) trust relation paths between u and v – two paths $\pi_{u,v}$ and $\pi'_{u,v}$ are vertex-disjoint if $\neg\exists x \in \pi_{u,v}$, $x \neq u, v$, so that $x \in \pi'_{u,v}$, two paths $\pi_{u,v}$ and $\pi'_{u,v}$ are edge disjoint if $\exists e \in \pi_{u,v}$, so that $e \notin \pi'_{u,v}$. We define a function $\tau_{path} : \mathcal{TP} \rightarrow [0, 1]$ computing the trust along the trust relation path according to Formula 1 by multiplying the trust values along the path (thus employing the trust discounting); we denote the computed trust value $\tau_{path}(\pi_{u,v})$ as $\tau_{\pi_{u,v}}$.

$$\tau_{path}(\pi_{u,v}) = \prod_{i=0}^{|\pi_{u,v}|} \{\tau_{x_i,x_{i+1}} \mid x_i \in \pi_{u,v}, x_0 = u, x_{|\pi_{u,v}|} = v\} \qquad (1)$$

The trust algorithm *GriTa* is quantified by a function $\tau_{gt} : V \times V \rightarrow [0, 1] \cup \perp$. We will explain the computation of the algorithm on the computation of a mediated trust value $\tau_{gt}(u, v)$ of the entity u in the entity v in the social trust network $stn(V, E, \tau)$.

If there is a trust relation (u, v), then, $\tau_{gt}(u, v) = \tau_{u,v}$. Otherwise, $|\pi_{u,v}| \geqq 2$ for arbitrary $\pi_{u,v} \in \mathcal{TP}$. In that case, we distinguish two types of trust: a trust of $x \in \pi_{u,v}$ in v, $(x, v) \in E$, and a trust of the entity u in the entity x which has trust relation to v. Both these types of trust satisfy Definition 1, the former – *direct trust* – is about x's trust in v (there exists trust relation $(x, v) \in E$) and the latter one – *trust in mediator* – is about how much the entity u trusts x's trust estimation of v. Trust in mediator is quantified by the function $\overline{\tau(u,x)} : V \times V \to [0,1]$, abbreviated as $\overline{\tau_{u,x}}$.

The trust algorithm GriTa has two phases. Phase 1 establishes a set of *trustworthy mediators* V^u_{truMed} selected in several steps. Phase 2, which is executed for the given entity u and the target entity v, selects the set of mediators $V^{u,v}_{Med}$ who have a direct trust in v and computes the final mediated trust value $\tau_{gt}(u,v)$ of the entity u in v based on the trust values $\overline{\tau_{u,x}}$ and $\tau_{x,v}$, where $x \in V^{u,v}_{Med}$.

Phase 1 - Establishing Trustworthy Mediators. Since $stn(V,E,\tau)$ can be huge, we suppose that the entity u defines $\kappa^u_{TruMed} \in \mathbb{N}$ specifying the desired $|V^u_{truMed}|$. Final trustworthy mediators V^u_{truMed} are selected in several steps – $V^u_{S_i}$ holds the mediators selected in Step i.

Firstly, in Step 1.1, $V^u_{S_{1.1}} = \{x \in V \mid \overline{\tau_{u,x}} > K\}$, where trust in mediator x, $\overline{\tau_{u,x}}$, is equal to the trust along the most trustworthy path as depicted in Formula 2. The function $\overline{\tau}$ comprehends trust as transitive, with the appropriate trust discount due to the use of Formula 1 in its computation. The time complexity of the algorithm implementing Formula 2 should be $O(|V|log|V| + |E|)$, because it can be computed using Dijkstra algorithm with a heap[2].

$$\overline{\tau_{u,v}} = max_{i=1}^{\#\pi^{\chi}_{u,v}}\{\tau_{\pi^i_{u,x}}\} \tag{2}$$

In Step 1.2, before entering Step 1.3, we set $V^u_{S_{1.2}}$ to contain at maximum $\overline{\kappa^u_{TruMed}} = \kappa^u_{supp}\kappa^u_{TruMed}$ entities $v \in V^u_{S_{1.1}}$ with the highest $\overline{\tau_{u,v}}$; $\kappa^u_{supp} \in \mathbb{N}$ is a parameter of the algorithm (see below). Therefore, Step 1.2 restricts the number of candidates on the trustworthy mediators, which lowers the complexity of the algorithm GriTa. Since Step 1.3 can throw out some entities $y \in V^u_{S_{1.2}}$, $\overline{\kappa^u_{TruMed}} \geqq \kappa^u_{TruMed}$.

Since the algorithm GriTa must be robust, the algorithm does not rely on one trust relation path with the highest trust computed according to Formula 2. Suppose that a malicious entity m tricks an entity $y \in V^u_{S_{1.2}}$, thus, $\tau_{y,m} = TH$. As a result, such a malicious entity can easily become part of $V^u_{S_{1.2}}$ during Steps 1 and 2. What is more, the malicious entity m can introduce trust relations to other malicious entities and, thus, polluting the set of mediators $V^u_{S_{1.1}}$ and $V^u_{S_{1.2}}$ with lots of malicious entities.

To avoid that situation, Step 1.3 further refines the set $V^u_{S_{1.2}}$: $V^u_{S_{1.3}} = \{x \in V^u_{S_{1.2}} \mid \#\pi_{u,x} \geqq \kappa_{supp}\}$. To compute $\#\pi_{u,x}$ we need to introduce an auxiliary network $g(V', E', cp) \in \mathcal{G}$, where V' is a set of vertices, E' set of edges, and $cp : E' \to 1$ defines a unit capacity for every edge of the network g. Then, according to Theorem 11.4 [3], network flow represented by a function $fl : \mathcal{G} \times V' \times V' \to \mathcal{R}^+$ launched on the network g, a source $a \in V'$, and a sink $b \in V'$ – $fl(g,a,b)$ – holds the maximum number of edge-disjoint paths between the vertices a and b in the network g [3]. We suggest using Edmond-Karp's algorithm for the computation of the network flow (the function fl), with a time complexity $O(|V||E|^2)$.

[2] Since the Dijkstra algorithm is searching "shortest" paths, the weights on the edges must be $1 - \tau_{x,y}$, $\forall(x,y) \in E$

When computing $\#\pi_{u,x}$, $\forall x \in V^u_{S_{1.2}}$, we cannot simply set $V' = V^u_{S_{1.1}}$, $E' = E_{|V^u_{S_{1.1}} \times V^u_{S_{1.1}}}$, and call $fl(g, u, x)$, because we are interested in the maximum number of **vertex**-disjoint trust relation paths. To construct V', E', and, thus, convert the task to searching the number of **edge**-disjoint trust relation paths, we use the approach suggested in the proof of Theorem 11.6 [3]: (1) $\forall v \in V^u_{S_{1.1}} \cup \{u\}$, we create two vertices $v^-, v^+ \in V'$; (2) $\forall v \in V^u_{S_{1.1}}$, we create one edge $(v^+, v^-) \in E'$; (3) from each trust relation $e = (w, z) \in E_{|V^u_{S_{1.1}} \times V^u_{S_{1.1}}}$, so that $\overline{\tau_{u,w}}\tau_{w,z} > K$ (the entity z must be trusted above the level K), we create the edge $(w^+, z^-) \in E'$ – as a result, each vertex $v^-/v^+ \in V'$ has only incoming/outgoing edges. Then, calling $fl(g, u^+, x^-)$ computes $\#\pi_{u,x}$.

Finally, V^u_{truMed} is set to be equal to $V^u_{S_{1.3}}$ containing at maximum κ^u_{TruMed} entities v with the highest $\overline{\tau_{u,v}}$. Thanks to Step 1.3, the computed $\overline{\tau_{u,v}}$, $v \in V^u_{truMed}$ is supported by at least κ_{supp} vertex-disjoint trust relation paths $\pi^i_{u,v}$, $1 \leqq i \leqq \#\pi_{u,x}$.

The time complexity of Phase 1 is $O(|V^u_{S_{1.2}}||V||E|^2)$ due to $|V^u_{S_{1.2}}|$ runs of the Edmond-Karp's algorithm.

Phase 2 - Computing Derived Trust Value. Phase 2, which is executed for the given entity u and the target entity v, selects the set of mediators $V^{u,v}_{Med}$, $V^{u,v}_{Med} = \{x \in V^u_{truMed} \mid (x, v) \in E\}$. If $|V^{u,v}_{Med}| \geqq \kappa^u_{minMed}$, where $\kappa^u_{minMed} \in \mathbb{N}$ defines the u's desired minimum number of mediators, the final mediated trust value $\tau_{gt}(u, v)$ of the entity u in v is computed according to Formula 3 as a weighted average over the trust values $\tau_{x,v}$ with the weights $\overline{\tau_{u,x}}$, $x \in V^{u,v}_{Med}$. Otherwise the entity's v trustworthiness cannot be judged, $\tau_{gt}(u, v) = \perp$.

$$\tau_{gt}(u,v) = \frac{\sum_{x \in V^{u,v}_{Med}} \overline{\tau_{u,x}}\tau_{x,v}}{\sum_{x \in V^{u,v}_{Med}} \overline{\tau_{u,x}}} \tag{3}$$

The time complexity of Phase 2 is $O(|V^{u,v}_{Med}|)$.

5 Extending Trust Algorithm GriTa

Since it may happen that $|V^{u,v}_{Med}| < \kappa_{minMed}$ in Formula 3, thus, the number of obtained trustworthy mediators is less than a certain threshold κ_{minMed}. To obtain more mediators, we analyze reputations of all entities (based on trust relations of all entities present in the social trust network $stn(V, E, \tau)$) and select top k mediators with the highest reputation and with a direct trust in v; $k = \kappa_{minMed} - |V^{u,v}_{Med}|$. Before outlining the extension of the algorithm GriTa with the reputation (Subsection 5.2), we need to explain the concept of reputation values (Subsection 5.1).

5.1 Reputation Values

Reputation value of the entity is computed according to function $r : \mathcal{V} \rightarrow [0, 1]$ using internally PageRank algorithm originally proposed in [17]. Computation of $r(v)$ is depicted in Formula 4 – $r(v)$ represents the likelihood that an entity randomly following the trust relations in the social trust network arrive at the entity v [17]. The probability is influenced by the reputation values of the entities $\{x \mid (x,v) \in E\}$, which have a trust relations to v and by the total number of other outgoing trust relations of x ($outdeg(x)$). Therefore, the more entities are pointing to the entity v and the less other entities are pointed from the entities pointing to v, the higher reputation the entity v has. Since even the imaginary entity randomly following the trust relations will eventually stop following, there is a probability d in Formula 4 that the entity will continue and the probability $1-d$ that the entity will choose random entity in the social trust network (see [17] for more details); typically, $d = 0.85$.

$$r(v) = \frac{(1-d)}{|V|} d \sum_{\{x \mid (x,v)\in E\}} \frac{r(x)}{outdeg(x)} \tag{4}$$

Since trust relations in our social trust network $stn(V, E, \tau)$ are weighted by the function τ, we modify the original formula for computing PageRank to support weights on the trust relations (see Formula 5).

$$r(v) = \frac{(1-d)}{|V|} d \sum_{\{x \mid (x,v)\in E\}} \frac{\tau_{x,v} r(x)}{\sum_{\{y \mid (y,v)\in E\}} \tau_{y,v}} \tag{5}$$

The computation of the reputation values $r(v)$ for all entities $v \in V$ outlined in Formula 5 is done in iterations; $r^i(v)$ denotes the reputation of the entity v after the iteration i. In the simplest method, initially, $\forall v \in V$, $r^0(v) = \frac{1}{|V|}$; afterwards, the computation of $r^{i+1}(v)$ is done according to Formula 6. Future will show, whether more powerful implementations of the PageRank algorithm, such as the the algorithm introduced in [2], should be used.

$$r^{i+1}(v) = \frac{(1-d)}{|V|} d \sum_{\{x \mid (x,v)\in E\}} \frac{\tau_{x,v} r^i(x)}{\sum_{\{y \mid (y,v)\in E\}} \tau_{y,v}} \tag{6}$$

The whole computation is finished, when, $\forall v \in V$, $|r^{i+1}(v) - r^i(v)| < \kappa_\epsilon$, for the given $\kappa_\epsilon \to 0$.

5.2 Trust algorithm GriTa with Reputation Support

In this section, we extend the way the mediators $V^{u,v}_{Med}$ are chosen in Phase 2 of the algorithm GriTa. Suppose that $k = (\kappa_{minMed} - |V^{u,v}_{Med}|) > 0$. To supplement the set $V^{u,v}_{Med}$ with k other mediators with high reputation and use them to compute $\tau_{gt}(u,v)$, we need to (1) convert reputation values to trust in mediator values represented by the function $\overline{\tau}$ and (2) select top k entities with the highest trust values $\overline{\tau} > K$ and with a direct trust in the entity v.

The trust in mediator value $\overline{\tau_{u,x}}$, $x \in V$, is determined based on the position of $r(x)$ among the β% of entities V with the highest reputation, $\beta \leqq 50$; details are in Table 3. For example, if the entity $x \in V$ has $r(x)$ among top 21% (but not among top 14%) of the reputations of all entities, $\overline{\tau_{u,x}} = 0.7$. Based on that, we define a set of *mediators with high reputation* $V^{u,v}_{repMed}$, $V^{u,v}_{repMed} = \{x \in V \mid (x,v) \in E \land \overline{\tau_{u,w}} > K\}$, where $\overline{\tau_{u,w}}$ is obtained from Table 3.

Table 3. Determining $\overline{\tau_{u,w}}$ based on the position of $r(x)$ among the β% of entities V with the highest reputation

$\beta\%$	$\overline{\tau_{\mathbf{u},\mathbf{v}}}$
$\beta \leqq 7$	0.9
$7 < \beta \leqq 14$	0.8
$14 < \beta \leqq 21$	0.7
$21 < \beta \leqq 28$	0.6
$28 < \beta \leqq 35$	0.5
$35 < \beta \leqq 42$	0.4
$42 < \beta \leqq 50$	0.3

Finally, we set $V^{u,v}_{Med} = V^{u,v}_{Med} \cup topK(V^{u,v}_{repMed})$, where $V^{u,v}_{Med}$ is the set of chosen mediators in Formula 3 and the function *topK* selects the top k ($k = (\kappa_{minMed} - |V^{u,v}_{Med}|) > 0$) entities $x \in V^{u,v}_{repMed}$ with the highest $\overline{\tau_{u,x}}$. If the condition $V^{u,v}_{Med} < \kappa_{minMed}$ still holds, the sufficient amount of mediators (trustworthy mediators plus mediators with high reputation) $V^{u,v}_{Med}$ to compute $\tau_{gt}(u,v)$ cannot be supplemented, thus, $\tau_{gt}(u,v) = \perp$.

6 Trust Model's Support for Activities

In previous sections, we discuss the Trust model including the trust algorithm GriTa computing τ_{gt} among arbitrary two entities in the social trust network $stn(V,E,\tau)$. This section discusses how the Trust model is applied when various activities specified in the separated document "Requirement analysis of the SoSIReCr portal" occurs.

We have identified the following activities which must be be supported by the introduced Trust model:

- Searching users/groups/projects with the similar professional profiles (R6.9 – R6.11)
- Searching the demanding profiles of users/groups (R6.13)
- Aggregation of professional profiles of users/groups/projects in the chosen region of the Czech Republic (R6.14)

6.1 Searching users/groups/projects with the similar professional profiles (R6.9 – R6.11)

When an entity $u \in V$ searches for the users/groups/projects with the similar professional profiles, the similarity metric for the found professional profiles of the entities v must be supported with the computed mediated trust values $\tau_{gt}(u, v)$ and the portal must enable resorting of the professional profiles according to the mediated trust values. Furthermore, the portal must support the possibility to show up only entities v with the $\tau_{gt}(u, v)$ above the certain threshold.

6.2 Searching the demanding profiles of users/groups (R6.13)

If an entity $u \in V$ finds the entity $v \in V$ publishing the demanding professional profile, the computed $\tau_{gt}(u, v)$ represents the trust value the entity u should accord to the demanded professional profile of the entity v.

6.3 Aggregation of professional profiles of users/groups/projects in the chosen region of the Czech Republic (R6.14)

The function aggregating professional profiles in the chosen region must enable incorporation of the mediated trust value $\tau_{gt}(u, v)$ between the entity u requesting the aggregation and the particular entity v, whose professional profile is being aggregated. See Subsection 6.1 for more details.

7 Properties of Trust Algorithm GriTa and Related Work

Ziegler and Lausen [20] present a categorization of trust algorithms (metrics) – they distinguish global and local trust algorithms. Global trust metrics, such as [17, 11], compute the *reputation* of entities in social trust networks. On the other hand, local trust metrics comprehend trust as a subjective opinion of the particular entity. The trust algorithm GriTa behaves like a local trust metric if sufficient amount of trustworthy mediators is found; otherwise, it selects the mediators with the highest reputation, thus, behaves like a global trust metric.

Apart from trust transitivity, Golbeck [7] particularizes three other properties of trust: *personalization*, *asymmetry*, and *composability*. The personalization is satisfied as long as sufficient amount of trustworthy mediators is found. Assymetry is satisfied always. Composability means that if there exist more trust relation paths between entities u and v, the derived trust value is based on more/all these path. Composability of the direct trust is ensured by Formula 3; to achieve full composability, the algorithm would have to ensure composability of the trust in mediators computed according to Formula 1. Full composability, due to its higher time complexity, will be considered in the future version of the Trust model according to the performance results of the current algorithm.

Kuter and Golbeck propose in [13] an algorithm SUNNY for trust inference in social networks supported by a measure of confidence in the computed trust value. They evaluated the algorithm on the FilmTrust network [7] and compared the algorithm with TidalTrust algorithm – SUNNY's average error was 6.5% lower, performing much more better for $p < 0.05$ in the standard two-tailed t-test. Nevertheless, the probabilistic derivation of the confidence values seems to be not trivial (unfortunately, no big O notation is given) and the resulting improvement is rather minor.

Ziegler and Lausen [20] proposed Appleseed trust metric (as an improved version of the Advogato[3] trust metric [14]) calculating trust for a collection of entities at once by energizing the selected entity and spreading the energy to other entities connected by trust relations. The problem of this algorithm is the normalization of the trust values - the more trust relations the entity defines, the less energy (trust) each target entity of these trust relations receives. The algorithm GriTa does not have this problem.

Guha et al. [8] introduced several ways of propagating trust in a social network. Except of *direct propagation* (use of trust transitivity), they propose other atomic propagations – *co-citation* (if $\tau_{u_1,v_1} > K$, $\tau_{u_1,v_2} > K$, and $\tau_{u_2,v_2} > K$, we can infer $\tau_{u_2,v_1} > K$), *transpose trust* (if $\tau_{u,v} > K$, then $\tau_{v,u} > K$), and *trust coupling* (if $\tau_{u_1,v} > K$, $\tau_{u_2,v} > K$, then $\tau_{u,u_1} > K$ implies $\tau_{u,u_2} > K$). The algorithm GriTa uses the transitivity of trust. Other atomic propagations are too vulnerable to malicious entities, which can easily simulate the prerequisites of these propagations and obtain an extra trust.

Furthermore, Guha et al. [8] and Ziegler and Lausen [20] dealt with the propagation of distrust and observed that the semantics of transitivity is not clear: if $\tau_{u,v} \leqq K$ and $\tau_{v,w} > K$, the derived trust value of u in w is rather low, although the only way to deduce the trust in the entity w is from the distrusted entity v. In the algorithm GriTa, we simply do not use mediators v with $\tau_{u,v} \leqq K$. To derive trust of u in w in the social trust network, we employ the PageRank-like algorithm computing reputation of entities. Other option would be to incorporate an algorithms for finding experts among entities [19, 4].

8 Architecture of the Trust Subsytem

The architecture of the Trust subsystem is depicted in Figure 3. The architecture is client-server; server is denoted as SNT_S, client as SNT_C.

A server must enable the operations as follows:

– Storing social trust network, in particular the explicit and implicit trust relations between entities together with the time interval (t_1, t_2) during which the given trust relations were valid. The particular storage mechanism (relational database, graph database) should be chosen according to the storage mechanism used for the portal
– Adding explicit trust relation (created by SNT_C) to the social trust network. This triggers the updating of implicit trust relations.
– Removing explicit trust relation (removed by SNT_C) from the social trust network, t_2 of the removed trust relation must be specified. This triggers the updating of implicit trust relations.
– Updating explicit trust relation (updated by SNT_C) in the social trust network – this operation is equal to removing the existing explicit trust relation

[3] http://www.advogato.org/

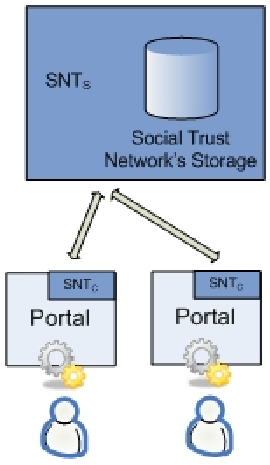

Fig. 3. Architecture of the Trust Subsystem.

currently updated and adding the new explicit trust relation (see above). This triggers the updating of implicit trust relations.

- Updating implicit trust relations based on the added/removed/updated explicit trust relation and Table 2. This operation should be fast and, thus, can be done as the requests on adding/removing/updating explicit trust relations arrives. If this shows as a bad assumption, a separated thread should be created to serve the incoming requests on the FIFO basis.
- Computing reputation values for all entities according to Formula 5. Reputation is always computed for the snapshot of the social trust network involving trust relations valid in the given time t – therefore, if some trust relations are updated/added/removed during the computation of the reputation, this fact must not influence the current computation of the reputation and will be included to the next computation of the reputation. Experiments must show, how often the reputation should be computed.
- Computing Phase 2 of the algorithm GriTa for every situation introduced in Section 6. This can trigger the computation of the set of trustworthy mediators V^u_{truMed} and the computation of reputation values, if not already computed.
- The set of trustworthy mediators V^u_{truMed} should be computed every time Phase 2 of the algorithm GriTa is executed for the entity u and V^u_{truMed} was not yet computed or was computed before more than $\kappa_{medLastComp}$ minutes. Since Step 1.3 of the algorithm GriTa (finding vertex-disjoint trust relation paths) can be computation intensive, it must run in a separated thread; before the computation of Step 1.3 finishes, Phase 2 uses $V^u_{truMedTemp}$, which is a set of trustworthy mediators computed according to Phase 1 of the algorithm GriTa for the entity u, however, with Step 1.3 omitted.

A client, which must be integrated with the portal's Web GUI, must support the operations as follows.

- Creating explicit trust relation (by associating one of the trust levels proposed in Table 1 with the chosen entity).
- Removing explicit trust relation (by disassociating one of the trust levels proposed in Table 1 with the chosen entity).
- Updating explicit trust relation (by associating different trust level proposed in Table 1 with the chosen entity).
- Supplying the activities introduced in Section 6 with the mediated trust values and reputation values.

Client is using web services to communicate with the server; based on the client's and server's operations, the communication layer must support the following requests:

- Retrieving the particular explicit/implicit trust relation
- Retrieving all explicit/implicit trust relations for the particular entity
- Adding/Removing/Updating the explicit trust relation
- Retrieving results of the algorithm GriTa
- Retrieving reputation of the particular entity/set of entities

9 Implementation Notes

The Trust subsystem should be implemented in Java. Every public class, interface, and public method must be documented, other methods and classes should be documented as well; documentation must be in English.

9.1 Interfaces in the Implementation

To support future adjustment of the algorithm GriTa's implementation, at least the appropriate interfaces covering the following subalgorithms of the algorithm GriTa must be designed:

- V^{u}_{truMed} computeTMed(u,$stn(V, E, \tau)$) computing trustworthy mediators according to Phase 1 in Section 4
- $topK(V^{u,v}_{repMed})$ computeHRepMed(u,v,k,$stn(V, E, \tau)$) computing k mediators with the highest reputation according to Section 5
- $\tau_{gt}(u, v)$ computeGriTa(u,v,$V^{u,v}_{Med}$,$stn(V, E, \tau)$) computing Phase 2 (the final mediated trust value $\tau_{gt}(u, v)$) according to Section 4, with the reputation extension in Section 5

Preferably, Steps 1 and 3 in Phase 1 of the algorithm GriTa should be implemented in separated methods whose implementations could be changed in the future. Similarly, the computation of the PageRank algorithm according to Formula 5 should be separated from the conversion of the reputation values to the trust values according to Table 3.

9.2 Collecting Data Behind Rankings in the Discussion Posts

Whenever an entity u adds a discussion post p to a discussion d on the entity w's profile, the post can be ranked by one to five stars by another entity v. If this happens, a new record must be stored – the record must involve the time t of the ranking, participated entities u, v, and w, the post p, discussion d, and the professional profiles of the entities u, v, and w valid in the time t. Such records will be than used in the analyses of expertness and experience of entities in the particular axes of the professional profiles (see Requirement 6.15 and 6.16 in the separated document "Requirement analysis of the SoSIReCr portal").

References

1. Donovan Artz and Yolanda Gil. A Survey of Trust in Computer Science and the Semantic Web. *Web Semant.*, 5(2):58–71, 2007.
2. Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast Incremental and Personalized PageRank. August 2010.
3. Adrian Bondy and U. S. R. Murty. *Graph Theory (Graduate Texts in Mathematics).* Springer, 3rd corrected printing. edition, December 2007.
4. John G. Breslin, Uldis Bojars, Boanerges Aleman-meza, Harold Boley, Lyndon Jb Nixon, Axel Polleres, and Anna V. Zhdanova. Finding Experts using Internet-based Discussions in Online Communities and Associated Social Networks. In *First International ExpertFinder Workshop*, 2007.
5. Yolanda Gil and Donovan Artz. Towards Content Trust of Web Resources. *Web Semant.*, 5(4):227–239, 2007.
6. Jennifer Golbeck. Trust on the world wide web: a survey. *Found. Trends Web Sci.*, 1(2):131–197, 2006.
7. Jennifer Ann Golbeck. *Computing and Applying Trust in Web-based Social Networks.* PhD thesis, College Park, MD, USA, 2005.
8. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *International World Wide Web Conference*, 2004.
9. Tom Heath. *Information-seeking on the Web with Trusted Social Networks from Theory to Systems.* PhD thesis, Milton Keynes, UK, 2008.
10. Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, March 2007.
11. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.
12. Tom Knap and Irena Mlýnková. Towards Topic-Based Trust in Social Networks. 7th International Conference on Ubiquitous Intelligence and Computing, 2010. (http://www.ksi.mff.cuni.cz/˜knap/wqam/tt10.pdf).
13. Ugur Kuter and Jennifer Golbeck. SUNNY: a new algorithm for trust inference in social networks using probabilistic confidence models. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1377–1382. AAAI Press, 2007.
14. Raph Levien. Attack-Resistant Trust Metrics. pages 121–132. 2009.
15. S. Marsh. Formalising Trust as a Computational Concept, 1994.

16. D. Harrison Mcknight and Norman L. Chervany. The Meanings of Trust. Technical report, University of Minnesota, Carlson School of Management, 1996.
17. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
18. Frank Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, February 2008.
19. Jun Zhang, Mark S. Ackerman, and Lada Adamic. Expertise networks in online communities: structure and algorithms. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 221–230, New York, NY, USA, 2007. ACM.
20. Cai-Nicolas Ziegler and Georg Lausen. Propagation Models for Trust and Distrust in Social Networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.