Computational Social Network Management in Crowdsourcing Environments

Florian Skopik, Daniel Schall, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology
Argentinierstraße 8/184-1, A-1040 Vienna, Austria
{skopik|schall|dustdar}@infosys.tuwien.ac.at

***Abstract*—Flexible interactions in complex social and service-oriented collaboration systems increasingly demand for automated adaptation techniques to optimize partner discovery and selection. Today, applications of complex service-oriented systems can be found in crowdsourcing environments. In such environments, collaborations are typically short-lived and strongly influenced by incentives and actor behavior. As actors prove their reliable and dependable behavior in jointly performed activities, they become increasingly considered as invaluable partners. A social network builds a strong basis to enable successful collaborations between crowd members. In order to keep track of the dynamics in such systems, it is inevitable to apply an autonomous approach to manage social network structures automatically using captured interaction data. Thus, we introduce an adaptation concept that accounts for emerging social relations based on varying interaction behavior of collaboration partners. We describe the foundational concepts for dynamic social link management in Web-based collaborations. We highlight major concerns of computational models in highly dynamic networks and deal with temporal aspects such as supporting the emergence of relations, efficient update mechanisms, and aging of relations.**

***Keywords*-computational social network management; emergence of social relations; service-oriented crowdsourcing**

I. Introduction

Over the past years, the Web has transformed from a pool of statically linked information to a people-centric Web. Various Web-based tools and services have become available enabling people to communicate, coordinate, and collaborate in a distributed manner. *Crowdsourcing* has emerged as an important paradigm in human problem solving techniques on the Web. More often than not, programs outsource tasks to humans which are difficult to implement in software. Applications range from enterprise environments [1] to open Internet based platforms such as Amazon Mechanical Turk (MTurk[1]). These online platforms distribute problem-solving tasks among a group of humans. Crowdsourcing follows the 'open world' assumption allowing humans to provide their capabilities to the platform by registering themselves as services. Some of the major challenges are monitoring of crowd capabilities, detection of missing capabilities, strategies to gather those capabilities, and tasks' status tracking [2].

Service-oriented architecture [3] (SOA) enables the design of applications that are composed from the capabilities of distributed services that are discovered at runtime. Unlike traditional SOA-based approaches, we consider complex service-oriented systems that are established upon the capabilities of human and software services [4]. The integration of human capabilities in a service-oriented manner is motivated by the difficulties to adopt human expertise into software implementations. Instead of dispensing with human capabilities, people handle tasks behind traditional service interfaces. In contrast to process-centric flows (top-down compositions), we advocate flexible compositions wherein services can be added at any time exhibiting new behavior properties. Hence, our service-oriented approach to the design and implementation of flexible collaboration networks enables the realization of versatile application scenarios. However, especially the involvement of and dependencies on humans as a part of flexible compositions has a major impact on all aspects of the system since dynamics and evolution are driven by software services and human behavior [5]. We propose an interaction mining and self-adaptation approach of service-oriented collaboration networks. In crowdsourcing environments, where people and services dynamically interact to perform activities, reliable and dependable behavior promotes the emergence of *social relations* and *trust* [6]. We argue that relations from a social perspective can neither be reliably identified in advance nor defined statically. They rather emerge dynamically upon interaction behavior of humans and services, and evolve over time. Sophisticated social network management models need to account for these properties to reflect real situations as close as possible. Moreover, in highly flexible environments, interaction behavior may alter quickly and, therefore, the underlying model has to be updated in sufficiently short cycles. However, in large-scale networks with potentially thousands of participants, updating relations in short intervals is not feasible due to limitations of computational power. Hence, relations have to be updated and altered selectively.

Here, we mainly address two issues of *computational social network models* resulting from interaction flexibility:

- *Efficiency* in terms of performance is realized by carefully selecting the most critical relations in a network to be refreshed in adaptive update cycles. Scheduling of updates fundamentally depends on the actors' interac-

[1]MTurk: http://www.mturk.com/

tion behavior, and the community's utility of frequent updates.

- *Effectiveness* in terms of functionality deals with the application of algorithms to let a model reflect the dynamically changing environment as close as possible. Our approach accounts for the different lifecycle phases of relations: *emergence-, update-, and aging phase*.

The remainder of the paper is organized as follows. Section II covers important related work. Section III shows a motivating scenario and deals with previous work that builds the basis for our approach. In Section IV we discuss novel concepts to cope with behavior dynamics and its temporal properties regarding emergence, update and aging of social relations. Then, we evaluate the applicability of the proposed model in Section V. Finally, Section VI concludes the paper and gives an outlook of our future work.

II. Background and Related Work

Adaptation of system behavior is introduced by establishing a cycle that feeds back environmental conditions. The MAPE cycle [7] is considered as one of the core mechanism to achieve adaptability through self-* properties. While autonomic computing allows for autonomous elements and applies these principles to distributed systems, current research efforts left the human element outside the loop. Based on the observed context of the environment, different adaptation strategies can be applied [8] to guide interactions between actors and actions to prevent inefficient use of resources and disruptions. Studies on distributed teams focus on human performance and interactions [9], [10] as well as *Enterprise 2.0* environments [11]. Socio-technical (mixed) service-oriented systems target flexible interactions and compositions of Human-Provided and Software-Based Services [4]. This approach is aligned with the vision of the Web 2.0, where people can actively contribute services [12]. Crowdsourcing is an important paradigm in distributed problem solving involving human actors in the Web. Some of the major challenges have been discussed in [2].

Trust in service-oriented systems has become a very important research area. SOA-based infrastructures are typically distributed comprising a large number of available services and huge amounts of interaction logs. Therefore, trust in SOA has to be managed in an automatic manner [13]. Depending on the environment, trust may rely on the outcome of previous interactions [14] and interest similarity [15], [16]. In our approach, metrics express social behavior influenced by the context in which collaborations take place [6]. For instance, *reciprocity* [14] is a concept describing that humans tend to establish a balance between provided support and obtained benefit from collaboration partners.

Many of today's social networks are declared by people manually in static lists. This work proposes the automated management of social relations based on interaction monitoring. Aging models for the WWW [17] describe common characteristic change rates of Web pages. Similar principles have been applied in social network analysis to update user profiles. For example, sliding window filters have been studied [18] to construct network approximations from interactions. Interaction behavior, interest similarities and joint group memberships, social relations, such as trust, can be predicted to some extent automatically [16], [19]. Here, we study the emergence of social relations not only from a trust perspective, but consider a multitude of social interaction metrics and behavioral styles.

III. Service-orientation in Crowds

We motivate our work with a concrete scenario and outline fundamental principles that have been discussed in earlier work, building the basis for this paper.

A. Scenario

Let us consider a mixed service-oriented crowdsourcing environment to introduce our concepts. The environment consists of professionals and experts who interact and collaborate by the means of information and communication technologies to perform work. The actors, i.e., the community network members, that are both humans and software services (i.e., autonomous agents), provide *help and support* on requests of each other. In such a mixed service-oriented environment actors have to register at a central community management service to become part of the network. Humans can register themselves by providing their profiles, including their education, employment status, certified skills and project experience. Services can be registered by their vendors or third party persons that offer information about service features and capabilities. In the described environment, network members perform activities. Activities are a concept to structure information in ad-hoc collaboration environments, including the goal of the ongoing tasks, involved actors, and utilized resources. They are either assigned from outside the community, e.g. belonging to a higher-level process, or emerge by identifying collaboration opportunities. Such opportunities are for instance writing a scientific paper because of having the required skills, and knowing *and trusting* the right

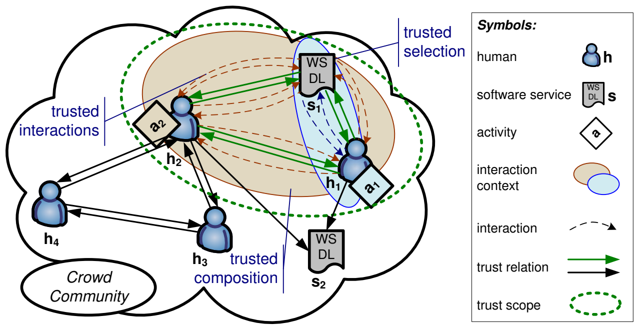


Figure 1. Mixed service-oriented crowdsourcing (color online).

supporting actors in a community (i.e., humans with the required knowledge, services that provide scientific data).

In the scenario depicted by Figure 1, the two humans h_1 and h_2 are the owners of activities a_1 and a_2 respectively. We assume activity a_1 is a software implementation activity and a_2 is a software testing activity in some higher-level software development process (not depicted here). The human h_1, requests support from the Web service s_1, that is a software implementation knowledge base, providing code examples and FAQs[2] about software implementation. The dashed arrows represent interactions (requests for support (RFSs)), such as retrieving articles from the knowledge base. Interactions are performed by traditional SOAP calls. Even the capabilities of humans are described by WSDL and communication takes place with SOAP messages (see Human-Provided Services [4]). The interaction context, described by activity a_1 (reflected by the blue-shaded area), holds information about involved actors, goal of the activity, temporal constraints (start, duration, milestones), assigned resources, planned costs, risk with respect to the whole software development process and so on. The detailed description is out of scope of this paper, however, we conclude, that an activity holistically describes the context of an interaction in our environment model.

Human h_2, the owner of activity a_2, performs his/her activity (software testing) jointly with the help of h_1 and s_1. For that purpose, s/he interacts with all activity participants, such as requesting help and assigning sub-activities. Trust emerges from interactions and is bound to a particular scope. Therefore, we aggregate interactions that occurred in a pre-defined scope, calculate metrics (numeric values describing prior interaction behavior), and interpret them to establish trust. The scope of trust is reflected by the green dashed ellipse in Figure 1. In the given scenario, the scope comprises trust relations between crowd members regarding help and support in 'software development'. So, regardless of whether interactions took place in context of activity a_1 or a_2, interactions of both contexts are aggregated to calculate metrics, because both interaction contexts adhere to the scope of software development. Finally, interaction metrics are interpreted using rules, and the degree of trust between each pair of previously interacting members is determined.

B. Fundamental Principles

We extensively studied flexible interactions [4], metrics [6], [20], and monitoring [5] of service-oriented collaboration environments in our previous work. We overview the main principles that are the basis for the proposed adaptive social network management model. We previously investigated models and techniques to cover:

- *Context-aware Interaction Models.* Usually, interactions on the Web are easily observable. In particular, service-oriented systems allow for context-aware logging of SOAP-based interactions.
- *Mining of Interaction Metrics from SOAP Logs.* Metrics describe the interaction and collaboration behavior of users (e.g., in terms of reliability, openness, contributing behavior) and can be determined through advanced log analysis.
- *Inference and Interpretation of basic Social Relations.* Rule engines and fuzzy inference approaches allow for a situation-based interpretation of interaction metrics.
- *Interaction Patterns spanning numerous Actors.* Interaction patterns support the emergence of social relations by introducing (i.e., connecting) previously unknown actors; for example, by enabling delegations of requests to third parties.

[2]frequently asked questions

IV. Computational Link Model

Reliable social trust relations in dynamic (crowd) environments typically cannot be statically defined, but evolve over time. An efficient social trust management model must frequently refresh its data to keep track of the real situation.

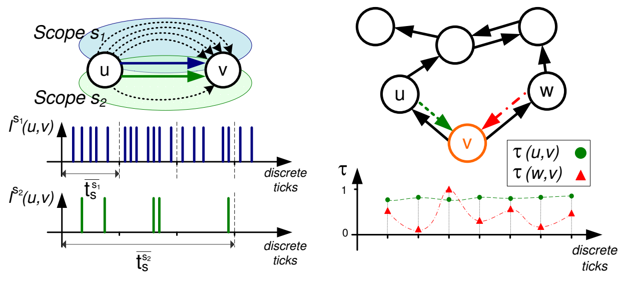
A. Challenges

We model the following fundamental lifecycle phases of social trust relations to account for their dynamic nature and temporal aspects: (i) *Emergence* deals with introducing new relations upon ongoing interactions; (ii) *Update* deals with refreshing existing relations based upon experiences made in recent interactions; (iii) *Aging* deals with degrading and deleting outdated relations.

Trust Emergence. Several concepts of link prediction exist to introduce new relations between actors. Recent research shows, that there is a strong dependency between interest similarities and trust [15], [19], [20]. Furthermore, there are approaches to recommend collaboration partners due to required expertise and reliable working styles. However, there is no evidence that collaboration partners will behave trustworthy according to predictions. Thus, in our model social trust relations are only built upon personal experiences from recent interactions and, hence, only if there is a sufficient amount of interactions to reliably infer trust. We enable the application of this concept through delegations, where unconnected actors start interacting within triadic closures [21] that support the emergence of trust relations.

Trust Update. Since the behavior of actors in a network may change due to various reasons, e.g., shift of interests, work overload, and search for new work opportunities, trust relations will alter as well. Hence, frequent synchronization with the real world is critical to computational trust models. For that purpose, the behavior of actors is sampled (i.e., observed through monitoring) in subsequent intervals and results are used to update the strength of social trust relations. A major challenge is to determine the appropriate sampling intervals (e.g., see also [22]). Figure 2 visualizes two fundamental challenges of trust update mechanisms:

- *Interaction Sparsity.* In different scopes s_1, s_2 occur varying types and amounts of interactions. Since a larger amount of interactions is needed to detect trends in an actor's behavior (e.g., responsiveness, availability), it is a challenging task to set the right size of sampling intervals ($\overline{t_s}$). Intervals that are too short prohibit reliably behavior analysis; however, if intervals are too long, sudden changes of behavior cannot be detected accordingly. The length of $\overline{t_s}$ mainly depends on the scope and the regular interaction behavior of actors therein.
- *Actor Uniformity.* The uniformity describes the consistency of an actor (i) towards the same partner over time; (ii) towards different interaction partners. In Figure 2(b) v behaves consistently trustworthy towards u, therefore, the level of trust $\tau(u, v)$ remains high over several sampling intervals. However, v alters dynamically his behavior toward w, and hence, w's trust in v changes rapidly over time[3]. Intuitively, in the second case of quickly changing behavior, smaller sampling intervals are required to capture v's behavior changes, while in the first case, the sampling interval to refresh already well-known constant behavior can be longer. Apart the actors' interaction behavior, external adaptation requirements may influence the determination of appropriate sampling intervals; e.g., in the case of pre-defined upper time limits to quickly react on sudden changes.



(a) Interaction sparsity. (b) Actor (non-)uniformity.

Figure 2. Challenges for interaction-based social relation update mechanisms in dynamic environments.

Trust Aging. If the amount of interactions between two actors falls below a certain threshold, or two actors completely stop interacting, trust relations undergo an aging process. Since in this phase no further evidence occurs for reliably interaction behavior, relations are not updated any longer. Therefore, trust relations (i.e., their strength) will degrade to a neutral state and are finally removed from the graph G. Intuitively, well established and consolidated long-term relations mature slower compared to fragile short-term relations. Hence, strengthened long-term relationships are able to bridge longer 'interaction gaps', while short-term relations disappear faster.

[3]Here, one could argue that oscillating interaction behavior is not trustworthy at all. However, we apply an optimistic point of view and appreciate recovery from unreliable behavior by increasing trust levels accordingly.

B. On the Emergence of Trust

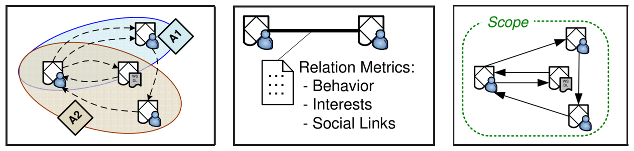
In contrast to a common security perspective, we define (social) trust to rely on the interpretation of previous collaboration behavior [23] and additionally consider the similarity of dynamically adapting interests [15], [20]. Especially in collaborative environments, where users are exposed to higher risks than in common social network scenarios [24], and where business is at stake, considering social trust is essential to effectively guide interactions [25]. Hence, we define trust as follows [14], [23], [26]:

> *Trust reflects the expectation one actor has about another's future behavior to perform given activities dependably, securely, and reliably based on experiences collected from previous interactions.*

Not only service interactions, but also human interactions may rely on SOAP (e.g., see Human-Provided Services [4] and BPEL4People [27]), which is the state-of-the-art technology in service-oriented environments, and well supported by a wide variety of software frameworks. This fact enables the adoption of various available monitoring and logging tools to observe interactions in service-oriented systems. Various metrics can be calculated from analyzing interaction logs. These relation metrics describe the links between actors by accounting for (i) recent interaction behavior, (ii) profile similarities (e.g., interest or skill similarities), (iii) social and/or hierarchical structures (e.g., role models). However, we argue that social trust relations largely depend on personal interactions.

We model a community of actors with their social relations as a directed graph, where the nodes denote network members, and edges reflect (social) relations between them. Since interaction behavior is usually not symmetric, actor relations are represented by *directed links*.

An outline of our approach to automatic interaction-based trust inference is depicted in Fig. 3. As motivated in the introduced use case, people interact to perform their tasks. This work is modeled as activities, that describe the type



(a) Interactions. (b) Relation metrics. (c) Scoped trust.

Figure 3. Trust emerging from interactions: (a) interaction patterns shape the behavior of actors in context of activities; (b) (semi-) automatic rewarding of behavior and calculation of interaction metrics; (c) trust inference in scopes by interpretation of metrics.

and goal of work, temporal constraints, and used resources. As interactions take place in context of activities (Fig. 3(a)), they can be categorized and weighted. Interaction logs are used to infer metrics that describe the relation of single actors (Fig. 3(b)), such as their behavior in terms of availability and reciprocity.

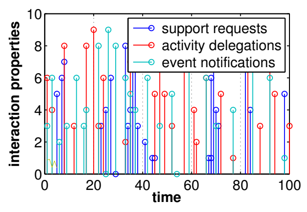
We support the diversity of trust by enabling the flexible aggregation of various interaction metrics that are determined by observing ongoing collaborations. Finally, available relation metrics are weighted, interpreted, and composed by a rule engine [6]. The result describes trust between the actors with respect to scopes (Fig. 3(c)). For instance, trust relations in a scope 'scientific dissemination' could be interpreted from interaction behavior of actors in a set of paper writing activities.

C. Adaptive Trust Update and Aging Models

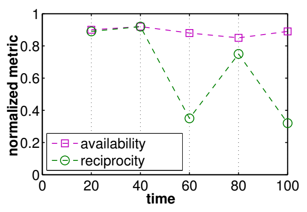
In our model, the selection of relations to be updated and update intervals rely on two influential factors (i) the variance of user behavior, reflected by the dynamics of interaction metrics, (ii) the sparsity of interaction, i.e., a certain amount of interactions is required to reliably determine interaction behavior. Note, for the initial establishment of social trust relations interactions are not mandatory, but relations can be introduced manually. This is a sufficient assumptions, as in real environments actors are selected based on recommendations or reputation values. However, once established, manually introduced relations are automatically updated by the system considering update and aging parameters. The advantage of manually introduced relations is the reduced effort when processing interaction logs. In this case, only interactions between actors who are already linked in the social network need to be handled.

Fundamental Update Mechanisms. Figure 4 summarizes the fundamental mode of operation of our temporal social trust management approach. Interactions from u to v (a), occurring between two sampling instants (in this example every 20 ticks), are utilized to calculate interaction metrics $M(u,v)$ (b). These metrics describe actor v's behavior toward u in scope s, and is inferred in consecutive sampling intervals $\overline{t_s}$; for instance, v's availability to u's requests and its reciprocity [14]. In the given example, the availability remains high, while v's reciprocity toward u is unsteady. We assume the level of trust $\tau^s(u,v)$ relies on both metrics. Therefore, in (c), recent trust, grounded in previous interaction behavior of v toward u in time interval $\overline{t_s}$, is inferred. This $\widehat{\tau}^s(u,v)$ is visualized in Figure 4(c) at the sampling instants.

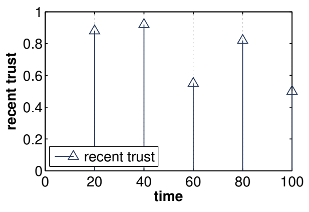
The strength/weight of evolving long-term relations τ_i^s (see Figure 4(d)) is updated periodically in successive time intervals t_i (e.g., days in mid-term collaboration scenarios), numbered with consecutive integers starting with zero. We denote trust values in scope s (context) calculated at time step i as τ_i^s. As the strength of relations is evolving over



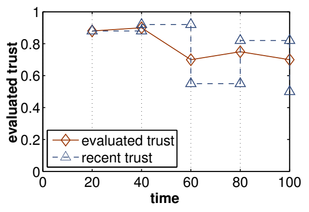
(a) Capture interactions.



(b) Calculate metrics.



(c) Interpret trust.



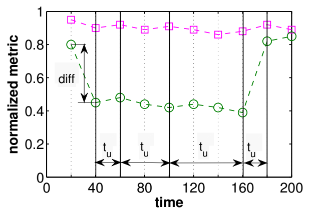
(d) Support trust evolution.

Figure 4. Illustrative example: update of trust relations based on captured recent interactions.

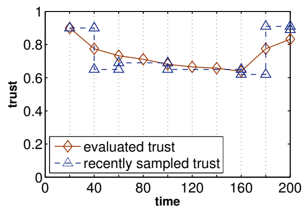
time, we do not simply replace old values, i.e., τ_{i-1}^s, with newer ones, but merge them accordingly. For this purpose we apply the concept of exponential moving average (EMA), to smoothen the sequence of calculated values as shown in Eq. 1. Using this method, we are able to adjust the importance of the most recent behavior (leading to $\widehat{\tau}^s$) compared to historical values. The smoothing factor $\alpha \in [0,1]$ can be dynamically adapted. The impact of the most recent values $\widehat{\tau}$ on well established long-term relations might be lower than on recently emerged and still fragile short-term relations. Long-term relations are normally based on large sets of previous experiences and sporadic short-term behavior changes, e.g., sporadic unreliability, may not have major impact. Indeed, this behavior is subjective, and our model can not dictate the application of this feature, but provides the means to cover such situations appropriately.

$$\tau_i^s = \alpha \cdot \widehat{\tau}^s + (1-\alpha) \cdot \tau_{i-1}^s \tag{1}$$

Adaptive Sampling. The fundamental approach simply updates τ_i at each time tick t_i, and the interval between instant t_i and t_{i+1} is constantly $\overline{t_s}$. However, in most real situations an adaptive sampling interval $\overline{t_s}$ is desired due to two reasons: (i) interaction sparsity, and (ii) actor (non-)uniformity (see Section IV). Intuitively, relations to erratic actors that change their behavior quickly and dynamically have to be updated more often, than the relations to actors with stable/consistent behavior. From a performance perspective, longer update cycles of stable connections allows the system to focus on unstable connections. Hence, while in the fundamental case we set the update interval in a particular scope s to $\overline{t_u^s} = \overline{t_s}$ (equal to the system sample interval), we introduce now an approach to adapt $\overline{t_u^s}$ dynamically within the limits according to Eq. 2.



(a) Trigger metric changes.



(b) Adaptive evaluation.

Figure 5. Illustrative example: adaptive update of trust relations through behavior triggers.

$$\overline{t^s_{u_{min}}} \leq \overline{t^s_u} \leq \overline{t^s_{u_{max}}} \tag{2}$$

Both limits are pre-configured and determined by the interaction sparsity. Furthermore, $\overline{t^s_{u_{min}}} = \lambda_1 \cdot \overline{t_s}$ and $\overline{t^s_{u_{max}}} = \lambda_2 \cdot \overline{t_s}$ and $\lambda_1 \leq \lambda_2$ for $\lambda_1, \lambda_2 \in \mathbb{N}$. The basic challenge is to find appropriate update intervals $\overline{t^s_u}$, in terms of efficiency and effectiveness of social trust management. Remember, although static relations do not need frequent updates, sudden behavior changes must not be neglected. The mode of operation of our adaptive approach is exemplarily depicted in Figure 5.

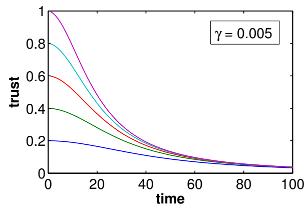
We interpret actor behavior, reflected by metrics M as a continuous 'signal' that is sampled from interactions in consecutive time intervals $\overline{t_s}$. Therefore, metrics reflect the changeability of an actor's behavior. Figure 5(a) shows the temporal evaluation of two interaction metrics. While the values of one metric are nearly constant over time, the other suddenly drops at time tick 40, remains low, and increases again at tick 180. We detect such rapid changes with precisely configured *event triggers*. Once sudden events are detected, such as the variance of the most recent values is above a threshold, or the number of unreplied requests is considerably high, an update operation is triggered (see tick 40). Then, when the metric values are stable, $\overline{t^s_u}$ is extended by one $\overline{t_s}$ in each update cycle. In our examples $\overline{t_s} = 20$, therefore, after tick 40 the next update intervals have the lengths $\overline{t_s}(= \overline{t^s_{u_{min}}})$, $2 \cdot \overline{t_s}$, and $3 \cdot \overline{t_s}$. However, at tick 180 a sudden behavior change is detected and link weights are sampled as soon as possible (instead of waiting a period of $4 \cdot \overline{t_s}$.

Typically rather simple and easily computable metrics that characterize the actor behavior and can efficiently capture behavior changes, are used to trigger update actions. While at least this set of metrics, is calculated at each $\overline{t_s}$, the larger amount of (typically more complex) metrics and finally trust values are refreshed only after adaptive intervals $\overline{t^s_u}$. This is visualized in Figure 5(b). Sampled trust $\widehat{\tau}$ is only captured at intervals $\overline{t^s_u}$. However, a temporal evaluation (Eq. 1) is still applied at each t_i (as in the fundamental approach), but based on the most recent $\widehat{\tau}$.

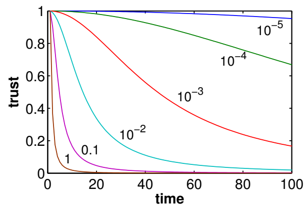
Trust Aging Model. As social trust relations in the real world degrade if people do not frequently interact, also relations in the computational model underlie an aging process. While it is intuitive that relations will become invalid over time, it is quite hard – if not impossible – to realistically reflect this aspect in a mathematical model. Our approach, as defined in Eq. 3, provides some parameters for tuning the aging process, while it is not too complex to be applied in real environments.

$$\tau^s_n = \tau^s_i \cdot e^{-(\tau^s_{n-1} \cdot \Delta t)^{2\gamma}} \tag{3}$$

The variable τ^s_i represents the latest determined value based on interactions in the update procedure that is degraded exponentially, configured by the decay factor γ ($\gamma \leq 1$). So, trust τ^s_n at time tick t_n is calculated by degrading τ^s_i depending on the time span $t_n - t_i$. The quality of computed relationships suffers if links are not periodically refreshed through new interactions. While immediately after updating a relation ($\Delta t = 0$) the strength is not altered ($\tau_n = \tau_i$), the aging process produces trust results asymptotic to zero with $\Delta t \to \infty$.



(a) Varying levels of trust w.



(b) Varying decay factors γ.

Figure 6. Illustrative example of trust aging from different strength levels w and for different decay factors γ.

Adaptive aging refers to the dynamic adaptation of γ, hence, the older a relation, the slower may be the applied aging process. Furthermore, as in real life, the decay of links can be comparably fast in the beginning, while the actual removal of relations takes longer time. The adaptation of the decay factor may depend on actors interaction consistency. The configuration of the aging model (see Figure 6) is still an open issue. On the one side domain experts could care for this based on best practice, on the other side there exist concepts that let systems adapt parameters autonomously [28] to optimize the aging process. However, we design the computational model to be flexible enough to cover various demands on temporal properties; e.g., sampling intervals ($\overline{t_s}$), impact of new values (α), decay factor (γ).

D. Computational Social Network Algorithm

Algorithm 1 formulates the emergence of new trust relations (Line 36), updates of existing ones (Line 10), and their aging in case no interactions take place between connected actors (Line 43). It manages links between a subset of nodes $N' \subseteq N$ in an existing trust network $G_T = (N, E_T)$ for a predefined set of scopes (depending on already existing links that need to be updated – see Line 5). In case the amount of

Algorithm 1 Update algorithm executed every tick t_i.

```
 1: /* access $G_I = (N, E_I)$ in interaction databases */
 2: /* access $G_T = (N, E_T)$ in social trust model */
 3: for each $u \in N'$ do
 4:     for each $v \in N'$ do
 5:         for each $s \in Scopes$(edge($E_T, u, v$)) do
 6:             if $|E_I(u, v)| > \vartheta_I^s$ then
 7:                 /* enough interactions to reliably infer trust */
 8:                 $e_\tau \leftarrow E_T(u, v)$
 9:                 if $\exists\ \tau^s \in e_\tau$ then
10:                     /* update of existing links scheduled */
11:                     if isUpdateScheduled($e_\tau, s$) then
12:                         /* previously scheduled update */
13:                         $M^s(u, v) \leftarrow$ calcMetrics($E_I(u, v), s$)
14:                         $\widehat{\tau}^s(u, v) \leftarrow$ inferTrust($u, v, M^s(u, v)$)
15:                         $\overline{t_u^s} \leftarrow$ getUpdateInterval($e_\tau, s$)
16:                         if $\overline{t_u^s} \leq \overline{t_{u_{max}}^s}$ then
17:                             scheduleUpdate($e_\tau, s, \overline{t_u^s} + \overline{t_s}$)
18:                         else
19:                             scheduleUpdate($e_\tau, s, \overline{t_{u_{max}}^s}$)
20:                         end if
21:                     else
22:                         /* trigger changing behavior */
23:                         $M_T^s(u, v) \leftarrow$ calcTriggers($E_I(u, v), s$)
24:                         if isUpdateTriggered($e_\tau, M_T^s(u, v)$) then
25:                             $M^s(u, v) \leftarrow$ calcMetrics($E_I(u, v), s$)
26:                             $\widehat{\tau}^s(u, v) \leftarrow$ inferTrust($u, v, M^s(u, v)$)
27:                             scheduleUpdate($e_\tau, s, \sigma_\tau, \overline{t_{u_{min}}^s}$)
28:                         else
29:                             /* stable behavior, no updates */
30:                             $\widehat{\tau}^s(u, v) \leftarrow \widehat{\tau}_{i-1}^s(u, v)$
31:                         end if
32:                     end if
33:                     /* smoothen trust values */
34:                     $\tau_i^s(u, v) \leftarrow$ update($\tau_{i-1}^s(u, v), \widehat{\tau}^s(u, v)$)
35:                 else
36:                     /* introduce new links */
37:                     $M^s(u, v) \leftarrow$ calcMetrics($E_I(u, v), s$)
38:                     $\tau_i^s(u, v) \leftarrow$ setInitialTrust($M^s(u, v)$)
39:                     addLink($e_\tau, s, \tau_i^s$)
40:                     scheduleUpdate($e_\tau, s, \overline{t_{u_{min}}^s}$)
41:                 end if
42:             else
43:                 /* if too few interactions */
44:                 $\overline{t_u^s} \leftarrow$ getUpdateInterval($e_\tau, s$)
45:                 if $\overline{t_u^s} \leq \overline{t_{u_{max}}^s}$ then
46:                     /* increase update intervals */
47:                     scheduleUpdate($e_\tau, s, \overline{t_u^s} + \overline{t_s}$)
48:                 else
49:                     /* age out existing relations */
50:                     applyAging($e_\tau, s$)
51:                 end if
52:             end if
53:         end for
54:     end for
55: end for
56: /* write back updated $G_T$ */
```

interactions to reliably infer behavior (and trust) is above a predefined threshold (ϑ_I^s depends on the 'usual' amount of interactions in a scope), new relations are introduced and existing ones updated respectively. New edges are added to G_T if a significant amount of interactions took place between two actors but no trust relations exist yet (Line 36). The level of trust (τ) is inferred from measured metrics (see [6] for details about rule-based trust inference) and updates are scheduled as soon as possible – still accounting for interaction sparsity in the given scope ($\overline{t_{u_{min}}^s}$). Updates are performed due to two events: (i) an update has been scheduled for a given relation; (ii) a rapid change in an actor's behavior has been triggered and thus, connecting links have to be updated to reflect this change in the model accordingly. In the first case (see Line 11), update cycles are extended up to $\overline{t_{u_{max}}^s}$ in order to optimize performance. Hence, for longer stable interaction behavior of actors, update intervals are increased.

However, if considerable sudden changes in behavior are detected (e.g., someone does not reply to requests anymore) (see Line 22), an immediate update is triggered and consecutive updates are performed in shorter intervals until stable behavior (trust levels) is detected again. If the amount of interactions drops below a given threshold ϑ_I^s, update intervals are increased to collect a sufficient number for reliable trust determination. However, if the update interval become too long ($> \overline{t_{u_{max}}^s}$), the previously described aging process is applied. Function `applyAging()` (Line 50) is implemented as Eq. 3 that continuously degrades trust links, and finally, removes an existing edge from the graph model.

Algorithm 1 is periodically executed to keep G_T fresh. The execution interval needs to be adapted to the inherent dynamics of the environment. Since the algorithm processes interaction logs and relations only for a subset N' of all nodes, computational effort can be distributed over several instances that handle only parts of the whole network G_T.

V. Evaluation and Discussion

Since we have not yet applied our approach in real large-scale environments, we do not have sufficient real testing data. Therefore, we generate artificial scale-free network structures that we would expect to emerge under realistic conditions in typical collaboration networks [29] to test and discuss our computational social trust model.

A. Experiment Setup

Collaboration Network Generation. We utilize the *preferential attachment model* of Barabasi and Albert [29] to create graphs with power-law distributed degrees depicted in Figure 7. These network structures are the basis to generate interaction logs that follow a realistic distribution among members. For a graph $G = (N, E)$, we generate in total $100 \cdot |E|$ interactions between pairs of nodes (u, v). In our experiments we assume that 80% of interactions take place between 20% of the most active users (reflected by hub nodes with high degree). Generated interactions have a particular type (support request/response, activity success/-failure notification) and timestamp, and occur in one of two abstract scopes. Through utilizing available interaction properties, we calculate three metrics (i) *availability* (amount of responded support requests), (ii) *interest similarity* (based on extracted tags from successfully finished activities), and (iii) *support reciprocity* (ratio of served to requested support). The actual strength (weight respectively) of a social trust relation is determined by combining and weighting these metrics with a rule based approach (see [6] for details). For all experiments, we calculate the following interaction metrics:

Interest Similarity $isim$. This metric determines the overlap of actor interests, which is an important measure to find motivated partners in the same interest area. We manage keywords used by actors u and v as interest profile vectors $\mathbf{p}_u$ and $\mathbf{p}_v$ respectively (see [20] for details), and determine the similarity of profiles through the cosine between their profile vectors (Eq. 4). The result is a value between 0 (no overlap) and 1 (full overlap).

$$isim(u,v) = cos(\mathbf{p}_u, \mathbf{p}_v) = \frac{\mathbf{p}_u \cdot \mathbf{p}_v}{|\mathbf{p}_u||\mathbf{p}_v|} \tag{4}$$

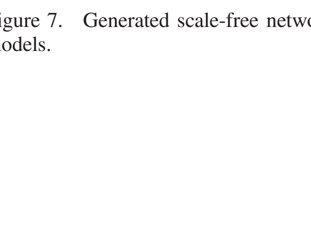
Reciprocity $recpr$. A typical social behavior metric is reciprocity [14] that here reflects the ratio between obtained and provided support in a community. Let $REQ(u,v)$ be the set of u's sent support requests to v, and $RES(u,v)$ the set of u's provided responses to v's requests. Then we define reciprocity in $[-1,1]$ as in Eq. 5; hence, 0 reflects a balanced relation of mutual give and take.

$$recpr(u,v) = \frac{|RES(u,v)| - |REQ(u,v)|}{|RES(u,v)| + |REQ(u,v)|} \tag{5}$$

Availability $avail$. This metric describes u's availability for v's requests, i.e. the amount of answered requests. The result of Eq. 6 is a value in $[0,1]$.

$$avail(u,v) = 1 - \frac{|REQ(v,u)| - |RES(u,v)|}{|REQ(v,u)|} \tag{6}$$

Model Setup. As described earlier, the computational model infers social trust by interpreting various measured metrics; here: $isim$ and $recpr$. Changing interaction behavior is triggered by varying availability ($avail$) of actors regarding requests from other members in the network.


(a) hierarchical (avg deg=2).

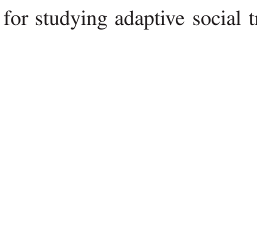

(b) democratic (avg deg=4).

Figure 7. Generated scale-free networks for studying adaptive social trust models.

This means that $avail$ is periodically sampled, while trust relations are updated based on $isim$ and $recpr$ only due to major changes of $avail$ (or the maximum update interval has been reached). We argue that these metrics are appropriate examples for reflecting reliable (i.e., trustworthy) behavior in typical activity-centric collaborations. In particular, for successful collaboration mutual interests are of importance, while also a cooperative behavior (expressed by support reciprocity) is highly rewarded. In contrast to that, in an emergency help and support environment (see [6]) fast and reliable response behavior is of paramount importance; thus, different metrics (responsiveness, success rate) denote trustworthy behavior there.

B. *Effectiveness of Adaptive Update Strategy*

We prove the advantages of selective and adaptive updates with several evaluations. For the following experiments, we set up a simulation environment as follows: We directly model different user behavior here to demonstrate the applicability of adaptive update intervals. In this round-based simulation the metrics $avail$, $recpr$, and $isim$ are modified for a fixed amount of actors. In particular, 5%, 10%, and 20% of (erratic) actors change in each round (with length $\overline{t_s}$) respective metric values randomly between 1% and 50%. We introduce $G_R = (N, E_R)$ which is a graph reflecting the reality, and modify metrics assigned to its edges $e_r \in E_R$. Social trust is managed in $G_T = (N, E_T)$ and its edges $e_\tau \in E_T$ updated by Algorihtm 1 according to changing metrics in G_R (reflects basic behavior sampling). The main goal of adaptive updates, compared to periodic intervals, is the reduction of update cycles due to performance reasons. However, by delaying updates a deviation (Eq. 7) between G_T and G_R is introduced that has to be kept to a minimum. The average deviation $dev(G_R, G_T)$ reflects the effectiveness of update models. The proposed update approach in Section IV has several tuning parameters. Among the most important ones are the settings of minimum/maximum update intervals, configured as $\overline{t^s_{u_{min}}} \leq \overline{t^s_u} \leq \overline{t^s_{u_{max}}}$; whereas $\overline{t^s_{u_{min}}} = \lambda_1 \cdot \overline{t_s}$ and $\overline{t^s_{u_{max}}} = \lambda_2 \cdot \overline{t_s}$ and $\lambda_1 \leq \lambda_2$. Hence, λ_2 allows to extend the scheduled updates of stable relations up to $\overline{t^s_{u_{max}}}$ and thus, to significantly reduce computational effort.

$$dev(G_R, G_T) = \frac{\sum_{e \in E} |\tau(e_r) - \tau(e_\tau)|}{|E|} \tag{7}$$

The introduced *global error* due to adaptive updates (compared to fixed interval updates) is expressed as the average $dev(G_R, G_T)$ in percent. Figure 8 depicts this error for different λ_2. In this experiment, the behavior trigger mechanism (compare Line 22 in Algorithm 1) has been deactivated. Instead, we decrease $\overline{t^s_u}$ by one $\overline{t_s}$ after each update operation. Hence, the lengths of future update intervals directly depend on the lengths of recent update intervals, but are only moderately influenced by sudden

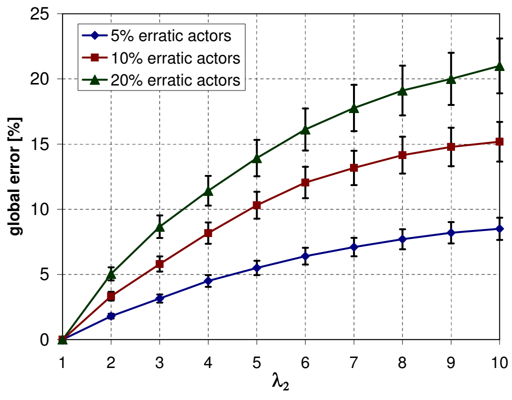


Figure 8. Deviation of trust values (global error) between simulated network and captured model for differently configured update strategies ($\lambda_1 = 1$).

behavior changes. It is demonstrated that even for small λ_2 considerable error rates are introduced. Since simulated behavior relies on various randomly changed metrics, error bars indicate the spread of results for multiple runs of this experiment. Although λ_1 determines $\overline{t^s_{u_{min}}}$, there is also an additional trigger mechanism that initiates immediate updates independent from $\overline{t^s_{u_{min}}}$ if actors change their behavior very quickly. With the trigger threshold ϑ_t the limit of tolerated behavior change without triggering an immediate update can be set. This threshold is defined as the deviation in percent of metric values in the interval $\overline{t_s}$. We trigger behavior changes by frequently observing the metric $avail$. With this trigger mechanisms, an upper limit of global error rate can be guaranteed, because rapid behavior changes (reflected in G_R) are detected and immediate updates of G_T performed. Hence, deviations are not added up over multiple sampling intervals (up to $\overline{t^s_{u_{max}}}$).

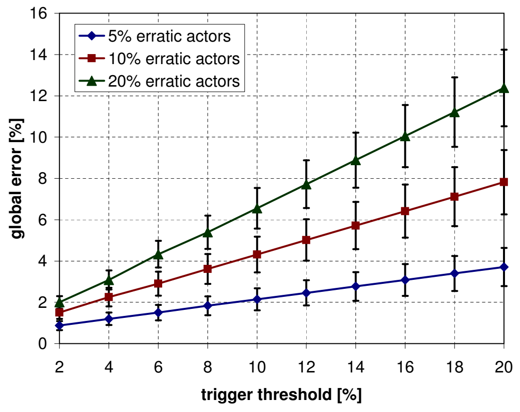


Figure 9. Deviation of trust values (global error) between simulated network and captured model for differently configured trigger thresholds ϑ_t ($\lambda_1 = 1$, $\lambda_2 = 5$).

Figure 9 visualizes that with the trigger mechanism in place, the global error rates can be considerably decreased. Typically, a higher number of erratic actors in the network still causes a higher average global error. The reason for that is a significant amount of actors who change their behavior slightly below the trigger threshold. Thus, a deviation of G_R to G_T is caused, but no updates triggered. However, setting a smaller λ_2 results in a smaller $\overline{t^s_{u_{max}}}$ and forces frequent updates; therefore, introduces an upper limit of global error rates over time. Since we have now demonstrated that we can keep the global error rate low, even when we apply adaptive updates (especially with a behavior change trigger in place), we demonstrate now the performance advantages. For that purpose, we utilize a generated graph G_R with $10\,000$ nodes and $20\,000$ edges (i.e., $d = 4$). In particular, we investigate the average amount of update operations per $\overline{t_s}$ for different λ_2. Higher λ_2 cause less frequent updates of relations. Note, updates of relations are not synchronous, i.e., all at the same point in time, but time instants are set for each edge individually in multiples of $\overline{t_s}$.

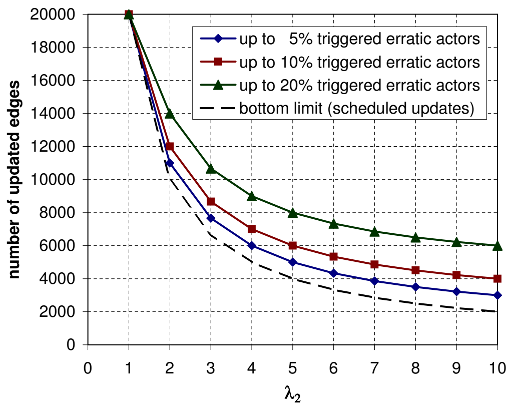


Figure 10. Number of processed edges after updating the trust model according to the simulated network ($|E| = 20\,000$, $\lambda_1 = 1$).

Theoretically, without adaptive updates and behavior triggers (i.e., $\lambda_1 = \lambda_2 = 1$), approximately $20\,000$ ($= |E|$) operations per $\overline{t_s}$ would be required to keep the example graph up-to-date with an error rate virtually equal to zero. However, since updates may be postponed until $\overline{t^s_{u_{max}}}$ if no rapid behavior changes are detected, the number of required update operations in G_T drops exponentially for higher λ_2, as shown in Figure 10. The dashed line visualizes the number of updated edges due to scheduled updates, even if actors do not change their behavior (then, all updates are performed in intervals of $\overline{t^s_{u_{max}}}$). The other lines show the upper limit of performed updates, i.e., the case that the set of relations with scheduled updates and relations with detected behavior changes do not overlap. Usually, the number of required updates is somewhere between these two limits.

Finally, Table I summarizes our results by comparing introduced global errors and required update operations; thus, demonstrating potential savings.

Table I
SUMMARY OF UPDATE STRATEGY EVALUATION ($\lambda_1 = 1$, $\vartheta_t = 10\%$, $amount\ of\ erratic\ actors = 10\%$).

λ_2	global error [%]	average number of updates
1	0	20 000
3	1.7	8 666
5	3.8	6 000
10	5.1	4 000

VI. CONCLUSION AND FURTHER WORK

In this paper we highlighted the application of the widely adopted MAPE approach for adaptations in complex interaction networks. Adaptation techniques, accounting for contextual constraints and emerging social relations (e.g., trust) are among the key research areas in flexible service-oriented collaboration environments. The evaluation of our model discovered important design issues, such as the mode of operation and configuration of dynamic update models. Our approach has important implications on adaptations in complex systems, because it reduces configuration burdens for the users and permits self-regulation of collaborations.

Our future work includes the deployment and evaluation of the implemented framework in the EU FP7 project COIN. There, business end-user evaluations will discover the usability of self-managed social networks in real environments. Furthermore, we plan to extend our work in the area of crowdsourcing with the focus on incentive schemes, price negotiation, and rewarding. As business is more and more performed online, the application of self-managed social network models, relying on collected interaction data, user actions, and personal profiles, in large-scale crowdsourcing environments seems to be a promising research field.

ACKNOWLEDGMENTS

This work is supported by the EU through the projects COIN (FP7-ICT-216256), and SM4ALL (FP7-ICT-224332).

REFERENCES

[1] M. Vukovic, "Crowdsourcing for enterprises," in *Congress on Services*, 2009, pp. 686–692.

[2] D. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, p. 75, 2008.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services - Concepts, Architectures and Applications*. Springer, October 2003.

[4] D. Schall, H.-L. Truong, and S. Dustdar, "Unifying human and software services in web-scale collaborations," *Internet Computing*, vol. 12, no. 3, pp. 62–68, 2008.

[5] H. Psaier, F. Skopik, D. Schall, and S. Dustdar, "Behavior monitoring in self-healing service-oriented systems," in *COMPSAC*. IEEE, 2010.

[6] F. Skopik, D. Schall, and S. Dustdar, "Modeling and mining of dynamic trust in complex service-oriented systems," *Inf. Syst.*, vol. 35, pp. 735–757, 2010.

[7] IBM, "An architectural blueprint for autonomic computing," *Whitepaper, 2005*, 2005.

[8] E. D. Nitto, C. Ghezzi, A. Metzger, M. P. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Autom. Softw. Eng.*, vol. 15, no. 3-4, pp. 313–341, 2008.

[9] P. A. Balthazard, R. E. Potter, and J. Warren, "Expertise, extraversion and group interaction styles as performance indicators in virtual teams," *DATA BASE*, vol. 35, no. 1, pp. 41–64, 2004.

[10] N. Panteli and R. Davison, "The role of subgroups in the communication patterns of global virtual teams," *IEEE Trans. Prof. Com.*, vol. 48, no. 2, pp. 191–200, 2005.

[11] J. Breslin, A. Passant, and S. Decker, "Social web applications in enterprise," *The Social Semantic Web*, vol. 48, pp. 251–267, 2009.

[12] C. Petrie, "Plenty of room outside the firm," *Internet Computing*, vol. 14, pp. 92–96, 2010.

[13] Z. Malik and A. Bouguettaya, "Reputation bootstrapping for trust establishment among web services," *Internet Computing*, vol. 13, no. 1, pp. 40–47, 2009.

[14] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation for e-businesses," in *HICSS*, 2002, p. 188.

[15] J. Golbeck, "Trust and nuanced profile similarity in online social networks," *ACM Tran. Web*, vol. 3, no. 4, pp. 1–33, 2009.

[16] Y. Matsuo and H. Yamamoto, "Community gravity: Measuring bidirectional effects by trust and rating on online social networks," in *WWW*, 2009, pp. 751–760.

[17] B. E. Brewington and G. Cybenko, "How dynamic is the web?" *Comp. Netw.*, vol. 33, no. 1-6, pp. 257–276, 2000.

[18] G. Kossinets and D. Watts, "Origins of homophily in an evolving social network," *American Journal of Sociology*, vol. 115, no. 2, pp. 405–450, 2009.

[19] C.-N. Ziegler and J. Golbeck, "Investigating interactions of trust and interest similarity," *Decision Support Systems*, vol. 43, no. 2, pp. 460–475, 2007.

[20] F. Skopik, D. Schall, and S. Dustdar, "Start trusting strangers? bootstrapping and prediction of trust," in *WISE*, 2009, pp. 275–289.

[21] D. Watts, *Six degrees: The science of a connected age*. W.W. Norton & Company, 2003.

[22] C. Domingo, R. Gavaldà, and O. Watanabe, "Adaptive sampling methods for scaling up knowledge discovery algorithms," *Data Min. Know. Disc.*, vol. 6, pp. 131–152, 2002.

[23] F. Skopik, D. Schall, and S. Dustdar, "Trustworthy interaction balancing in mixed service-oriented systems," in *SAC*, 2010, pp. 801–808.

[24] C. Dwyer, S. R. Hiltz, and K. Passerini, "Trust and privacy concern within social networking sites: A comparison of facebook and myspace," in *Americas Conference on Information Systems*, 2007.

[25] M. J. Metzger, "Privacy, trust, and disclosure: Exploring barriers to electronic commerce," *Journal on Computer-Mediated Communication, 2004,*, vol. 9, no. 4, 2004.

[26] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications Surveys and Tutorials, 2000,*, vol. 3, no. 4, 2000.

[27] A. Agrawal et al., "Ws-bpel extension for people (bpel4people), version 1.0," 2007.

[28] V. Bryl and P. Giorgini, "Self-configuring socio-technical systems: Redesign at runtime," *Int'l Trans. on Syst. Science and App.*, vol. 2, no. 1, pp. 31–40, 2006.

[29] A. Reka and Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, 2002.