Information Systems 35 (2010) 735–757

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Modeling and mining of dynamic trust in complex service-oriented systems

Florian Skopik *, Daniel Schall, Schahram Dustdar

Distributed Systems Group, Vienna University of Technology, Argentinierstr 8/184-1, 1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 11 March 2010
Accepted 12 March 2010
Recommended by: D. Shasha

Keywords:
Collaborative environment
Service-orientation
Social trust
Interaction patterns
Flexible composition
Crowdsourcing

ABSTRACT

The global scale and distribution of companies have changed the economy and dynamics of businesses. Web-based collaborations and cross-organizational processes typically require dynamic and context-based interactions between people and services. However, finding the right partner to work on joint tasks or to solve emerging problems in such scenarios is challenging due to scale and temporary nature of collaborations. Furthermore, actor competencies evolve over time, thus requiring dynamic approaches for their management. Web services and SOA are the ideal technical framework to automate interactions spanning people and services. To support such complex interaction scenarios, we discuss mixed service-oriented systems that are composed of both humans and software services, interacting to perform certain activities. As an example, consider a professional online support community consisting of interactions between human participants and software-based services. We argue that trust between members is essential for successful collaborations. Unlike a security perspective, we focus on the notion of social trust in collaborative networks. We show an interpretative rule-based approach to enable humans and services to establish trust based on interactions and experiences, considering their context and subjective perceptions.

1. Introduction

The way people interact in collaborative environments and social networks on the Web has evolved in a rapid pace over the last few years. Services have become a key-enabling technology to support collaboration and interactions. Pervasiveness, context-awareness, and adaptiveness are some of the concepts that emerged recently in service-oriented systems. A system is not only designed, deployed, and executed; but rather evolves and adapts over time. This paradigm shift from closed systems to open, loosely coupled Web services-based systems requires new approaches to support interactions [1].

We present a novel approach addressing the need for flexible discovery and involvement of experts and knowledge workers in distributed, cross-organizational collaboration scenarios. Experts register their skills and capabilities as Human-Provided Services (HPS) [2] using the very same technology as traditional Web services to join a professional online help and support community. This approach is inspired by *crowdsourcing* techniques following the Web 2.0 paradigm. People can contribute HPSs to offer their skills to a broad number of Web users, service compositions, and enterprises that need to have on-demand access to experts. In such communities, not only humans participate and provide services to others, but also autonomous software agents and semantic Web services with sophisticated reasoning capabilities. A mixed service-oriented system comprises human- and software services that can be flexibly and dynamically composed to perform various kinds of activities. Therefore, interactions in such a system do not only span humans, but also software services. Recently, *trust* has

* Corresponding author. Tel.: +43 1 58801 58418; fax: +43 1 58801 18491.
E-mail addresses: skopik@infosys.tuwien.ac.at, florian.skopik@gmx.at (F. Skopik), schall@infosys.tuwien.ac.at (D. Schall), dustdar@infosys.tuwien.ac.at (S. Dustdar).

0306-4379/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2010.03.001

been identified as a beneficial concept in large-scale networks [3,4]. Considering trust relations when selecting people for communication or collaboration, services to be utilized, and resources to be applied leads to more efficient cooperation and compositions of human- and software services [5]. In contrast to many others, we do not discuss trust from a security perspective. In this work we share the view of [6] that is related to how much humans or other systems can rely on services to accomplish their tasks.

Unlike several other systems in the agent domain, e.g., see [7], we follow a centralized trust management approach [5]. In SOA, central registries and logging facilities are common mechanisms. Applying them avoids various issues, such as the malicious manipulation of interaction data and dishonesty regarding recommendations. Moreover, some trust inference mechanisms are only applicable if the participants of the network have a global view on the system. However, on the other side, a centralized approach may raise privacy issues that have to be considered in the system design. In this paper, we present the following key contributions:

- *Social and behavioral trust model*. We define a trust model that relies on interaction dynamics, supporting wide personalization by accounting for user preferences, and discuss its realization in the introduced use case.
- *VieTE framework*. We outline VieTE (Vienna Trust Emergence), a modular framework that supports the management of trust in SOA-based environments. In particular, we introduce key implementation aspects, such as interaction mining, and *Web of Trust* provisioning.
- *Evaluation and discussion*. Since our work is not only theoretical, but closely coupled to SOA technology, we evaluate various functional and non-functional aspects of VieTE and its trust model.

The paper is organized as follows. In Section 2, we introduce the Expert Web case showing the need for flexible expert discovery and involvement. Our novel approach is based on *social* trust. We introduce trust concepts in collaborative environments in Section 3. Section 4 details the concept of interaction-based behavioral trust which will be the basis for our trust inference model. Trust can be based on different metrics whose meaning is highly subjective. In Section 5, we show our trust model established on fuzzy set theory. The trust model manages context-dependent trust between actors, i.e., humans and services, emerging from interactions. The subsequent Section 6 formalizes the fundamental trust model relying on captured and interpreted interactions. Successful, thus highly trusted network members, are valuable collaborators. However, overload due to large amounts of work represent bottlenecks. In Section 7, we present a balancing approach to prevent inefficient interactions. Our architecture is implemented on-top of SOA and Web services. We show the implementation details of the system in Section 8. Section 9 deals with evaluations to test the performance of the presented system as well as effectiveness of balancing algorithms. Finally, we discuss related work in the area of SOA, social trust, and flexible interactions models in Section 10 and conclude the paper in Section 11.

2. Service-oriented collaborations

In virtual communities, where people dynamically interact to perform activities, reliable and dependable behavior promotes the emergence of trust. As collaborations are increasingly performed online, supported by service-oriented technologies, such as communication-, coordination-, and resource management services, interactions have become observable. By monitoring and analyzing interactions, trust can be automatically inferred [1,7–9]. In contrast to manual rating approaches for mainly static communities, automatic inference is well-suited for complex networks with short-running interactions between potentially thousands of rapidly changing network members.

We motivate our work with a scenario showing discovery of experts and flexible interaction support as depicted in Fig. 1. In this use case, a higher level process model may be composed of single tasks assigned to responsible persons, describing the steps needed to

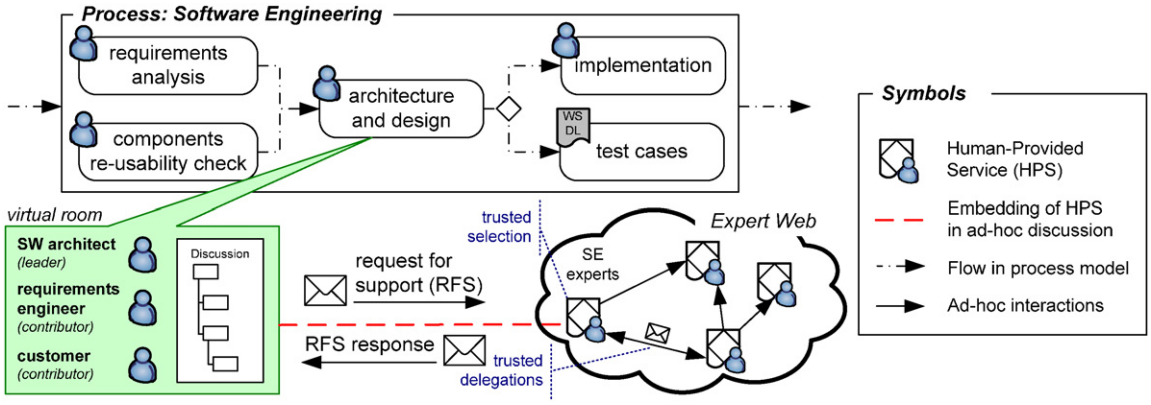

Fig. 1. Service-oriented large-scale collaboration in the *Expert Web*.

produce a software module. After finishing a common requirements analysis, and in parallel a reusability check of existing software artifacts produced in related projects, a software architect designs the actual software framework. The implementation task is carried out by a software developer, and additionally software test cases are generated with respect to functional properties (e.g., coverage of requirements) and non-functional properties (e.g., performance and memory consumption). We assume that this task is deployed in a global enterprise spanning multiple departments and locations. Thus, the single task owners in this process exchange only electronic files and interact by using communication tools. While various languages and techniques for modeling such processes already exist, for example BPEL, we focus on another aspect in this scenario: *interactions with trusted experts*. A language such as BPEL demands for the precise definition of flows and input/output data. However, even in carefully planned processes with human participation, for example modeled as BPEL4People activities [10], ad hoc interactions and adaptation are required due to the complexity of human tasks, people's individual understanding, and unpredictable events. In Fig. 1, the software architect receives the requirement analysis document from a preceding step. But if people have not yet worked jointly on similar tasks, it is likely that they need to set up a meeting for discussing relevant information and process artifacts. Personal meetings may be time and cost intensive, especially in cases where people belong to different geographically distributed organizational units. Various Web 2.0 technologies, including forums, Wiki pages and text chats, provide well-proven support for online-work in collaborative environments.

Several challenges remain unsolved: (i) If people, participating in the whole process, are not able to solve problems by discussion, who should be asked for support? (ii) How can experts be flexibly involved in ongoing collaborations? (iii) What are influencing factors for favoring one expert over others. (iv) How can we support trusted interactions in such dynamically changing environments and how can this situation be supported by service-oriented systems?

Traditionally, discovering support is simply done by asking third persons in the working environment, the discussion participants are convinced they are able to help, namely *trusted experts*. In an environment with a limited number of people, persons usually tend to know who can be trusted and what data have to be shared in order to proceed with solving problems of particular nature. Furthermore, they easily find ways to contact trusted experts, e.g., via phone or e-mail. In case requesters do not know skilled persons, they may ask friends or colleagues, who faced similar problems before, to recommend experts. The drawbacks of this approach are that people need extensive knowledge about the skills of colleagues and internal structures of the organization (e.g., the expertises of people in other departments). Discovering support in such a manner is inefficient in large-scale enterprises with thousands of employees and not satisfying if an inquiry for an expert becomes a major undertaking. Today's communication and collaboration technologies cannot fully address the mentioned challenges because many existing tools lack the capability of managing and utilizing *dynamic trust*.

The Expert Web: We propose the Expert Web, consisting of connected experts that provide help and support in a service-oriented manner. The members of this Expert Web are either humans, such as company employees offering help as online support services, or software services encapsulating knowledge bases. Such an enterprise service network, spanning various organizational units, can be consulted for efficient discovery of available support. Users, such as the engineer or drawer in our use case, send requests for support (RFSs). The users establish trust in experts' capabilities based on their response behavior (e.g., availability, response time, quality of support). This trust, reflecting personal positive or negative experiences, fundamentally influences future selections of experts. As in the previous case, experts may delegate RFSs to other experts in the network, for example, when they are overloaded or not able to provide satisfying responses. Following this way, not only users of the enterprise service network establish trust in experts, but also trust relations between experts emerge.

3. Communication, coordination, and composition

3.1. Social trust in collaborations

In contrast to a common security perspective, social trust refers to the interpretation of previous collaboration behavior [1] and may additionally consider the similarity of dynamically adapting interests [11,12]. Especially in collaborative environments, where users are exposed to higher risks than in common social network scenarios [13], and where business is at stake, considering social trust is essential to effectively guide interactions [14]. Hence, we define trust as follows (see also [1,8,9]):

> 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.

This definition includes several key characteristics that need to be supported by a foundational trust model:

- Trust reflects an expectation and, therefore, cannot be expressed objectively. It is influenced by subjective perceptions of the involved actors.
- Trust is context dependent and is basically valid within a particular scope only, such as the type of an activity or the membership in a certain team.
- Trust relies on previous interactions, i.e., from well-proven previous behavior a prediction of the future is inferred.

We strongly believe that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments. However, trust between human and software services is emerging based on interactions. Interactions, for example, may be categorized in terms of success (e.g., failed or finished) and importance.

Therefore, a key aspect of our approach is the monitoring and analysis of interactions to automatically determine trust in mixed service-oriented systems. We argue that in large-scale SOA-based systems, only automatic trust determination is feasible. In particular, manually assigned ratings are time-intensive and suffer from several drawbacks, such as unfairness, discrimination or low incentives for humans to provide trust ratings. Moreover, in the mentioned mixed system, software services demand for mechanisms to determine trust relations to other services. Much research effort has been spent on defining and formalizing trust models (for instance [7,9,15,16]). Although most of these models are closely related, e.g. in terms of concepts for recommendation and reputation, we add the following novel contributions.

Personalized trust inference: A fundamental characteristic of trust is its subjective perception. Humans have different requirements to establish trust to others. Therefore, we use a rule-based system, relying on fuzzy set theory that allows each participant of the network to define his/her own rules and influencing factors to establish trust. Instead of a 'hard-wired' logic to determine trust, we enable participants to model their individual *trust perception*, e.g., their optimistic and pessimistic views.

Multi-faceted trust: We support the diversity of trust by enabling the flexible aggregation of various interaction metrics that are determined by observing ongoing collaborations. Furthermore, data from other sources, such as human profiles and skills, as well as service features and capabilities may influence the trust inference process.

Compositional trust: The majority of today's trust models in the agent domain, such as typical buyer–seller scenarios, deal with the establishment of trust between exactly two entities. In contrast to that, we focus a compositional perspective, and study trust in group formation processes and compositions of services. In our environment, we understand compositions not from a structural perspective with pre-defined interaction paths (e.g., as in BPEL), but from a dynamic point of view, where members of the network select interaction partners flexibly [17].

3.2. The cycle of trust

Previously, we introduced a conceptual approach for determining trust based on interactions: the *cycle of trust* [5]. This cycle, adopting the MAPE concept [18], consists of four phases, which are *M* onitor, *A* nalyze, *P* lan and *E* xecute. Periodically running through these four phases establishes a kind of environmental feedback control, and therefore allows to adapt to varying circumstances. Applied in our environment, we are able to infer trust dynamically during ongoing collaborations. In the *monitoring phase* the trust management system observes interactions between humans and services, including their types, context and success. In the *analyzing phase* interactions are used to infer trust relationships. For this purpose, interaction metrics are calculated and interpreted using personal trust rule sets that depend on the purpose of and situation for trust determination. The following *planning phase* covers the set up of collaboration scenarios, including user activities and human-, and service compositions, taking the inferred trust relations into account. The *execution phase* provides support to enhance the execution of planned collaboration, including observing activity deadlines, checking the availability of actors, and compensation of resource limitations. The interactions of actors are observed in the execution phase; and the loop is closed.

4. From interactions to social trust

In this paper, we demonstrate the inference of trust depending on captured collaboration data considering individual trust perceptions. Conceptually we follow a three layer approach (Fig. 2), realizing trust emergence concepts that support our motivating scenario.

Interaction layer: On the bottom layer logging and analyzing of interactions take place. From atomic interactions, more meaningful and aggregated interaction metrics are extracted in subsequent time intervals by the means of message correlation and pattern detection. Domain-specific interaction metrics are determined in pre-configured scopes.

Personal trust layer: On the middle layer interaction metrics are combined and weighted individually ('interpreted') by applying configured rules, and a fundamental trust network is established. These personal trust relations reflect the individual trust perception of the actors.

Trust projection layer: On the top-layer potential future trust relations are predicted where no personal trust has been established yet.

While the interaction- and trust metrics on the first two layers are calculated offline in fixed subsequent time intervals (due to potentially high computational effort for personalized trust relations in large networks), trust projection on the top layer is performed dynamically when needed. In the following we discuss the three layers in detail focusing on personal trust.

4.1. Interaction layer

The identified key concepts on that layer are (i) harnessing diverse available collaboration data, (ii) enabling and capturing various types of interactions in SOA-based mixed systems environments, (iii) accounting for context models associated with these interactions, and

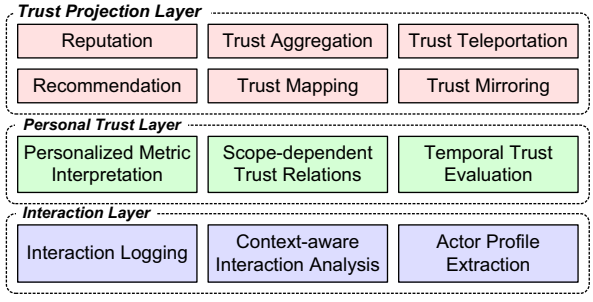

Fig. 2. Layered trust emergence approach.

(iv) defining trust scopes to allow a rule-based inference of trust from observed interactions.

4.1.1. Collaboration data

Recently, various trust models have been published relying on various mathematical concepts, e.g. probability [19–21] or reasoning rules [22]. Most works make no assumption about the data used to determine trust, so trust models are completely decoupled from the data underneath. In particular, many approaches account only for one-dimensional manual user feedback (rating) and categorize interactions only in cooperative and defective ones. However, the support of automatic inference demands for observable evidence of trust. Therefore, the large part of used data comes from observing interactions (see the research area of complex event processing[1]). Besides interactions, there are more data sources in collaborative and social networks that can be used to express the diversity of trust, and are utilized for some higher level trust projection concepts.

Finally, we identified the following sources, common in most large-scale SOA-based networks, such as described in the motivating use case:

- *Interactions*: Interactions provide evidence about the success of previous collaboration encounters. We categorize in fundamental interactions, such as e-mail traffic, instant messaging, VoIP communication and SOAP/REST-based service invocations; and in higher-level interactions, mostly relying on fundamental interactions, but annotated with their semantic meaning, e.g., file exchange, report submission, and task delegation.
- *Profiles*: Profiles contain valuable information about actors. Human profiles are about professional background, job position, skills, and expertises; service profiles can contain vendor information, features, and capabilities. Similarities of profiles are utilized by higher-level trust concepts, including trust mirroring and teleportation (detailed in the following sections). Profiles may be entered completely manually, or are (partly) determined dynamically based on interactions. An example for inferring expertise from activity involvements can be found in [23].
- *Structural relations and hierarchies*: Knowledge about memberships in groups, roles of humans or services, joint activities and projects can be used to extend the notion and perception of trust. For instance, in a business environment someone may require that potentially trusted partners are members in the same team or are employed by a certain organization.
- *Manually declared relations*: Nowadays, standardized technologies for specifying friend (buddy) networks (FOAF,[2] XFN[3]) are used. This enables, people on the one side to define trust relationships explicitly, but on the other side also distrust ('foes'). Using such explicit commitments may override and fine-tune automatically inferred relations.

Capturing all these information may raise privacy concerns. However, neither content of interaction messages is stored nor semantic analysis is performed. We follow a pure structural approach, which makes extensive use of metadata instead of the actual message content. Furthermore, after analyzing interactions, logs can be deleted, and there remain only higher level metrics. Data are stored and managed by a centralized architecture, so there is no need to propagate sensitive data through the network (as in peer-to-peer networks).

4.1.2. Context-aware interaction observation

Our approach considers two different levels of interactions. On the bottom level we capture *basic interactions*, including fundamental exchanges of e-mail messages, VoIP calls, instant messages, and basic Web services invocations via SOAP or REST. In most cases there is no possibility to determine the semantic meaning of these interactions, i.e., the reason for initiating a Skype call. It can be only observed if a call is accepted and its duration. However, on the top level, *Web service enabled interactions*, using dedicated tools for delegating activities, performing periodic reports, and requesting help and support, provide more information on the reason of interactions and nature of collaboration.

Furthermore, in mixed service-oriented systems, consisting of humans and software services, we distinguish interactions according to the type of interacting entities. Our framework—VieTE—accounts for the following types of interactions: (i) *human–human*, including instant messaging and e-mail via dedicated services with integrated interaction sensors; (ii) *human–service* typically service invocations via SOAP or REST interfaces, using an external access layer that intercepts and captures messages; (iii) *service–human*, e.g., reminder service or meeting scheduling service, notifying humans about events; (iv) *service–service*, in typical service compositions, e.g., modeled with BPEL.

4.1.3. Interaction metrics and scopes

Interaction metrics are calculated by observing (monitoring) interactions and further analysis. Therefore, metrics describe the interaction behavior of actors, either humans or services, in a mixed service-oriented system. Such metrics are, for instance, their responsiveness (e.g., measured by an average response time), the reliability in responding to requests, the ratio of performed to delegated tasks, or the variance in delivering periodic status reports.

However, these interaction metrics are of course valid only in particular situations. For example, interaction behavior varies depending on the risk actors are facing, or the benefit they are receiving. Depending on the environment, several external factors may influence the interaction behavior of actors, such as their motivation, interest or expertise. For example, in the outlined help and support environment in Section 2, people might be more responsive in their dedicated expertise areas than in topics outside their interests.

[1] http://complexevents.com/
[2] Friend-Of-A-Friend http://xmlns.com/foaf/0.1/.
[3] XHTML Friends Network http://www.gmpg.org/xfn/11.

Therefore, we introduce the notion of *scopes*. In general, scopes describe what activities are relevant for determining certain interaction metrics. All interactions within relevant activities are taken into account for metrics determination. Scopes are defined specifically for a domain and depending on business areas. In the previous Expert Web use case, for instance, one scope can define that all software implementation and software testing activities should be considered, to describe someone's collaboration behavior in the area of software engineering. Other scopes could define to account for interactions regarding management activities of team leaders (e.g., delivering status reports, delegating tasks to team members etc.), or aggregate interactions from risky activities only.

The definition of scopes is supported in two different forms:

- *Tag-based scopes*: Scopes are represented as lists of key-words (tags). All interactions within activities whose descriptions incorporate these keywords, are taken into account for metrics calculation.
- *Activity-based scopes*: Scopes are determined by matching constraints on explicitly defined activities, e.g., matching activity type, a minimum team size to indicate mass collaboration, or a maximum risk.

All interactions that take place in the context of matching activities are considered for interaction metrics calculation. In case more than one tag is set and constraints defined, respectively, there will be interaction contexts (i.e., activities) that match only partly (e.g., only two of three tags). Then the impact of interactions on a metric is weighted based on the degree of match. More information on observing, logging, aggregating, and analyzing interactions in mixed systems, as well as calculating metrics have been studied in [5,23,24].

4.2. Personalized trust inference

We model the network of humans and services with their trust relations as a directed graph $G=(V,E)$, where the `vertices` V denote the members of the network, and `edges` E reflect their trust relations in between. General profiles of network members are attached to the vertices. Furthermore, both vertices and edges are annotated with various `metrics` that describe collaboration behaviors of network members and their relationships. A `community` comprises a subset of vertices (and references to their connecting edges). Fig. 3 visualizes this model.

We distinguish the following classes of metrics:

- *Interaction metrics* (subsets of `MEdge`) describe the interaction behavior as explained before, such as an actor's responsiveness and reliability in distinct scopes.
- *Similarity metrics* (subsets of `MEdge`) provide information about skill-, feature-, or expertise similarities, depending on the type of actors.
- *Trust metrics* (subsets of `MEdge`) are interpreted from interaction- and similarity metrics, e.g., personal trust, symmetry of trust relations (bidirectional trust) and trust trends in certain time intervals.
- *Collaboration metrics* (`MCollaboration`) are bound to a user, and describe independent from collaboration partners someone's previous experiences, such as collected expertise by performing activities, and behavior, e.g., reciprocity [9]. Furthermore, edge metrics can be aggregated to calculate collaboration metrics; for instance, an average value of someone's responsiveness or availability.
- *Group metrics* (`MGroup`) provide information about average values and distribution of vertex- and edge metrics in a community, therefore, they are a valuable mean to determine a metric value relative to others in the same group.

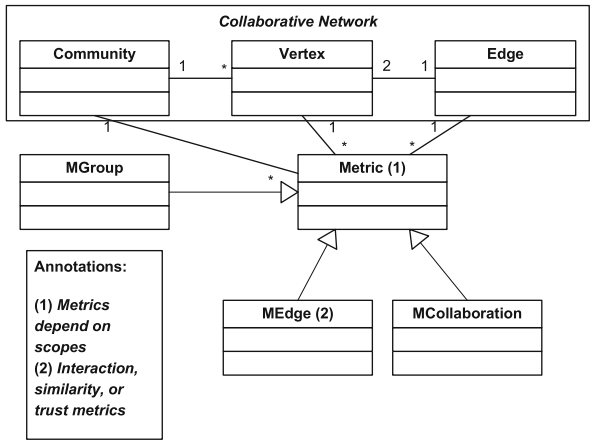

Fig. 3. Social and collaborative network model.

Users of the system, i.e., the network members, specify rules for evaluating calculated interaction metrics to trust. For this purpose, we utilize an approach based on fuzzy set theory (see more details in the next section) that (i) enables users to express their rules in almost natural language (similar to a domain specific language), and (ii) offers elegant and efficient mechanisms to aggregate fuzzy expressions of trust.

We decided to build a rule-based system, instead of a certain analytical model, because of the flexibility of the environment. Humans and services (i.e., service vendors) should be able to define their personal rules that have to be satisfied to establish trust in others. For instance, let us assume the members of the network want to describe influencing factors for establishing trust in a software engineer. There are various factors on that trust may rely: (i) based on formal skills provided by profiles including certificates such as university degrees, (ii) based on collected experience and previous success (e.g., performed activities of different types in the scope of software engineering), (iii) based on particular interaction behavior, such as support reliability and quality in the previously introduced Expert Web use case), or (iv) accounting for capabilities and behavior in related scopes of software engineering.

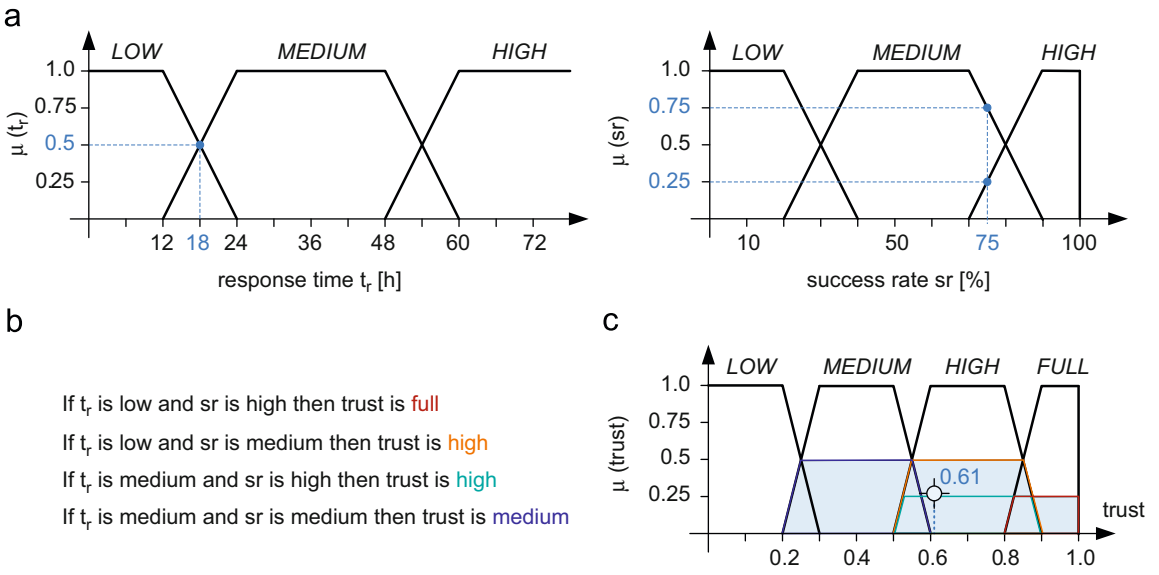

Fig. 4. An example showing fuzzy trust inference. Applied interaction metrics are response time t_r=18 h and success rate sr=75%: (a) definition of membership functions and fuzzified interaction metrics; (b) four applied fuzzy rules following the max–min inference and (c) defuzzification by determining the center of gravity.

4.3. Trust projection layer

In large-scale networks with thousands of humans and services, each member interacts only with a small amount of potential partners leading only to a small portion of personal trust relations from each member's point of view. Therefore, several concepts have been introduced to predict not existing relations, e.g., recommendation by the means of trust propagation, and reputation by the means of trust aggregation. We extend this list by three novel concepts, based on the similarities of actors, their trust perceptions, and the situations described by context data.

- *Trust mapping* deals with using trust relations established in other, but to some extent similar scopes (e.g., related expertise areas).
- *Trust mirroring* [12], implies that actors with similar profiles (interests, skills, community membership), tend to trust each other more than completely unknown actors.
- *Trust teleportation* [12] rests on the similarity of human or service capabilities, and describes that trust in a member of a certain community can be teleported to other members. For instance, if an actor, belonging to a certain expert group, is trusted because of his distinguished knowledge, other members of the same group may benefit from this trust relation as well.

5. Fuzzy set theory for trust inference

Fuzzy set theory, developed by Zadeh et al. [25], and fuzzy logic emerged in the domain of control engineering, but are nowadays more and more used in computer science to enable lightweight reasoning on a set of imperfect data or knowledge. The concept of fuzziness has been used earlier in trust models [26–28], however, to our best knowledge not to enable a personalized interpretation of trust from larger and diverse sets of metrics, calculated upon observable interactions. As fuzzy inference is a key mechanisms of our trust model, we introduce the fundamental definitions in this section. There exists various further literature on fuzzy set theory, for instance [29].

Zadeh et al. [25] defined a *fuzzy set* A in X ($A \subseteq X$) to be characterized by a membership function $\mu_A(x) : X \mapsto [0,1]$ which associates with each point in X a real number in the interval [0,1], with the value of $\mu_A(x)$ at x representing the 'grade of membership' of x in A. Thus, the nearer the value of $\mu_A(x)$ to 1, the higher the grade of membership of x in A. When A is a set in the ordinary sense of the term, its membership function can take only two values ($\mu_A(x) : X \mapsto \{0,1\}$, Eq. (1), according as x does or does not belong to A. Thus, in this case $\mu_A(x)$ reduces to the familiar characteristic function of a set A.

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \quad (1)$$

Eq. (2) depicts an example definition of a membership function $\mu_A(x)$ describing a fuzzy set. This membership function is part of the linguistic variable 'responsiveness' highlighted in Fig. 4(a), left side.

$$\mu_A(x) = \begin{cases} 0 & \text{if } 0 \le x < 12 \\ \frac{x}{12} - 1 & \text{if } 12 \le x < 24 \\ 1 & \text{if } 24 \le x < 48 \\ -\frac{x}{12} - 5 & \text{if } 48 \le x < 60 \\ 0 & \text{else} \end{cases} \quad (2)$$

Two or more fuzzy sets, describing the same characteristic (i.e., metric), can be merged to a *linguistic variable*.For instance in Fig. 4(a), the linguistic variable 'responsiveness' is described by three fuzzy sets: high, medium, and low.

The definition of linguistic variables (and the their single membership functions, respectively), has to be performed carefully as they determine the operation of the reasoning process. Linguistic variables are defined either for the whole community, or for groups, and even single network members, by:

- A domain expert, using his experience and expertise. However, depending on the complexity of the rules and aggregated metrics continuous manual adjustments are needed (especially when bootstrapping the trust system).
- The system itself based on knowledge about the whole community. For instance, the definition of a 'high' skill level is determined by the best 10% of all network members in certain areas.
- The users based on individual constraints. For example, a 'high' skill level from user *u*'s point of view starts with having more than the double score of himself.

Let X_A and X_B be two feature spaces, and sets that are describes by their membership function μ_A and μ_B, respectively. A *fuzzy relation* $\mu_R(x_A, x_B) : X_A \times X_B \mapsto [0,1]$ describes the set X, whereas $\mu_R(x_A, x_B)$ associates each element (x_A,x_B) from the cartesian product $X_A \times X_B$ a membership degree in [0,1]. Fuzzy relations are defined by a rule base (see example in Listing 1), where each rule, as shown in Eq. (3), comprises a premise p (condition to be met) and a conclusion c:

IF p **THEN** c (3)

Listing 1. Given the linguistic variables *response time* t_r, *success rate* sr, and *trust* τ, with the membership functions as defined in Fig. 4, we provide this rule base to the fuzzy engine.

```
if t_r is low    and sr is high   then τ is full
if t_r is low    and sr is medium then τ is high
if t_r is low    and sr is low    then τ is low
if t_r is medium and sr is high   then τ is high
if t_r is medium and sr is medium then τ is medium
if t_r is medium and sr is low    then τ is low
if t_r is high   and sr is high   then τ is medium
if t_r is high   and sr is medium then τ is low
if t_r is high   and sr is low    then τ is low
```

Approximate reasoning by evaluating the aforementioned rule base, needs some fuzzy operators to be defined [25]: **OR**, **AND**, and **NOT**.

$$A \textbf{ OR } B \equiv A \cup B \equiv \mu(x) = \max(\mu_A(x), \mu_B(x)) \quad \text{for } x \in X \tag{4}$$

$$A \textbf{ AND } B \equiv A \cap B \equiv \mu(x) = \min(\mu_A(x), \mu_B(x)) \quad \text{for } x \in X \tag{5}$$

$$\textbf{NOT } A \equiv \mu(x) = 1 - \mu_A(x) \quad \text{for } x \in X \tag{6}$$

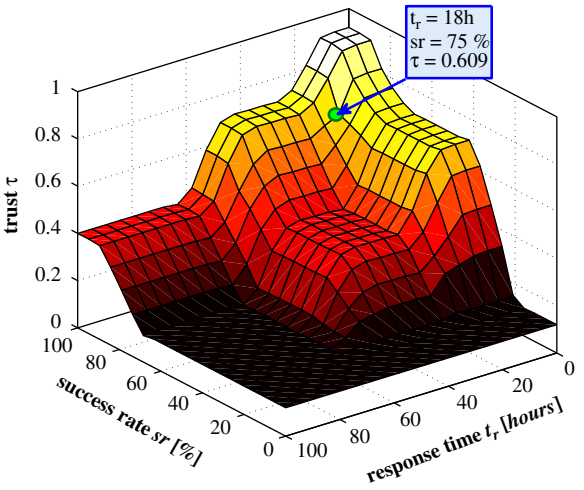

Fig. 5. Trust values after defuzzification for all possible input values of t_r and sr.

The *defuzzification* operation [30] determines a discrete (sharp) value x_s from the inferred fuzzy set X. For that purpose all single results obtained by evaluating rules (see Fig. 4(b)) are combined, forming a geometric shape. One of the most common defuzzification methods is to determine the center of gravity of this shape, as depicted in the example in Fig. 4(c). In general, center of gravity defuzzification determines the component x of x_s of the area below the membership function $\mu_x(x)$ (see Eq. (7)).

$$x_s = \frac{\int_x x \cdot \mu_x(x) \cdot dx}{\int_x \mu_x(x) \cdot dx} \tag{7}$$

Fig. 5 depicts possible trust values after defuzzification for metrics t_r and sr when applying membership functions defined in Fig. 4 and the rule base in Listing 1.

6. Trust model definitions

The trust model manages context-dependent trust between actors, i.e., humans and services, emerging from interactions (see Fig. 7(a)) that are captured and interpreted. A trust relation is always asymmetric, i.e., a directed edge from one vertex to another one in G. We call the trusting actor the *trustor* u (the source of an edge), and the trusted entity the *trustee* v (the sink of an edge). Analyzed *interactions* are any kind of communication, coordination or execution actions initiated by u regarding v. The *context* of interactions reflects the situation and reason for their occurrences, and is modeled as activities. Activities, as presented in [5] and shortly discussed before, describe work-relevant context elements. When interactions are interpreted, only a minor subset of all describing context elements is relevant within a *trust scope*. In the motivating use case of this paper, such a trust scope may describe the expertise area.

6.1. Fundamental trust model

Available metrics are processed by individually configured fuzzy (*event*)-*condition*-*action*-rules. These rules

define conditions to be met by metrics M for interpreting trustworthy behavior, e.g., 'the responsiveness of the trustee must be *high*' or 'a trustworthy software programmer must have collected at least *average* experiences in software integration activities'. Rules reflect a user's trust perception, e.g., pessimists may demand for stricter trustworthy behavior, than optimists.

On top of metrics, the *confidence* $c^s(u,v) \in [0,1]$ of u in v in scope s is determined. This confidence upon available interaction-, collaboration-, and similarity metrics $M(u,v)$ that describe the relationship from u to v, represents recent evidence that an actor behaves dependably, securely and reliably. Besides highly dynamic interaction metrics, actor profiles P may be considered during calculation, e.g., a human actor's education or a service's vendor. The function Ψ_c^s (Eq. (8)) evaluates u's fuzzy rule set $R_c(u)$ to determine confidence c in scope s in his collaboration partners (e.g., v). This confidence value is normalized to [0,1] according to the evaluation results of the rule base:

$$c^s(u,v) = \Psi_c^s(u, M(u,v), P(v), R_c(u), s) \tag{8}$$

The *reliability of confidence* $\rho(c^s(u,v)) \in [0,1]$, ranging from totally uncertain to fully confirmed, depends mainly on the amount of data used to calculate confidence (more data provide higher evidence), and the variance of metric values collected over time (e.g., stable interaction behavior is more trustworthy; see later about temporal evolvement). The function Ψ_ρ^s (Eq. (9)) determines the reliability ρ of confidence $c^s(u,v)$ relying on utilized metrics in $R_c(u)$. As the determination of reliability can be quite complex (considering temporal trends and variances of metrics), and the additional personal setup of this measure could be very demanding for the end-users, we let a domain expert configure a global reliability measure that accounts for metrics in $R_c(u)$ of respective network members:

$$\rho(c^s(u,v)) = \Psi_\rho^s(u, M(u,v), P(v), R_c(u), s) \tag{9}$$

Our engine infers *personal trust* $\tau^s(u,v) \in [0,1]$ by combining confidence with its reliability (see operator $\otimes$ in Eq. (10)). This can be performed either rule-based by attenuating confidence respecting reliability, or arithmetically, for instance by multiplying confidence with reliability (as both are scaled to the interval [0,1]). Since trust relies directly on confidence that is inferred by evaluating personal rules, an actor's personal trust relation in this model indeed reflects its subjective criteria for trusting another actor:

$$\tau^s(u,v) = \langle c^s(u,v), \rho(c^s(u,v)), \otimes \rangle \tag{10}$$

We introduce the *trust vector* $\vec{T}^s(u)$ to enable efficient trust management in the *Web of Trust*. This vector is combined of single personal trust relations (outgoing edges of a vertex in G) from an actor u to others in scope s (Eq. (11)):

$$\vec{T}^s(u) = \langle \tau^s(u,v), \tau^s(u,w), \tau^s(u,x), \dots \rangle \tag{11}$$

The *trust matrix* $\mathfrak{T}^s$ comprises trust vectors of all actors in the environment, and is therefore the adjacency matrix of the mentioned trust graph G. In this matrix, as shown in Eq. (12) for four vertices $V=\{u,v,w,x\}$, each row vector describes the trusting behavior of a particular actor ($\vec{T}^s$), while the column vectors describe how much an actor is trusted by others. If no relation exists, such as self-connections, this is denoted by the symbol $\perp$:

$$\mathfrak{T}^s = \begin{pmatrix} \perp & \tau^s(u,v) & \tau^s(u,w) & \tau^s(u,x) \\ \tau^s(v,u) & \perp & \tau^s(v,w) & \tau^s(v,x) \\ \tau^s(w,u) & \tau^s(w,v) & \perp & \tau^s(w,x) \\ \tau^s(x,u) & \tau^s(x,v) & \tau^s(x,w) & \perp \end{pmatrix} \tag{12}$$

In cases where actors define their personalized trust inference rules, the *trust perception* $p_\tau^s(u)$ represents the 'trusting behavior' of u, i.e., its attitude to trust others in scope s. The absolute value of $p_\tau^s(u)$ is not of major importance, but it is meaningful to compare the trust perceptions of various actors. Basically, this is performed by comparing their rule bases for trust inference (Eq. (13)), e.g., if actors account for the same metrics, or if they are shaped by optimism or pessimism. Therefore, more similar rules means more similar requirements for trust establishment. The application of trust perception becomes clear when discussing the trust projection concepts, such as weighting received recommendations based on the similarity of the recommender's trust perception:

$$sim_{percep}(p_\tau^s(u), p_\tau^s(v)) = sim(R_c^s(u), R_c^s(v)) \tag{13}$$

6.2. Temporal evaluation

Personal trust $\tau^s(u,v)$ from u in v is updated periodically in successive time intervals t_i, numbered with consecutive integers starting with zero. We denote the personal trust value calculated at time step i as τ_i^s. As trust is evolving over time, we do not simply replace old values, i.e., τ_{i-1}^s, with newer ones, but merge them according to pre-defined rules. For this purpose we apply the concept of exponential moving average,[4] to smoothen the sequence of calculated trust values as shown in Eq. (14).

$$\tau_i^s = \alpha \cdot \Delta\tau_i^s + (1-\alpha) \cdot \tau_{i-1}^s \tag{14}$$

With this method, we are able to adjust the importance of the most recent trust behavior $\Delta\tau^s$ compared to history trust values τ^s (smoothing factor $\alpha \in [0,1]$). In case, there are no interactions between two entities, but an existing trust relation, the reliability of this trust relation is lowered by a small amount each evaluation interval. Therefore, equal to reality, trust between entities is reduced stepwise, if they do not interact frequently:

Fig. 6(a) shows an example of applied EMA. The dashed line represents the trustworthiness of an actor's behavior, i.e., $\Delta\tau_i^s$, for the i th time interval, calculated independently from previous time intervals. In this extreme situation an actor behaves fully trustworthy, drops to zero, and behaves trustworthy again. Similar to reality, EMA enables us to memorize drops in recent

[4] http://www.itl.nist.gov/div898/handbook/

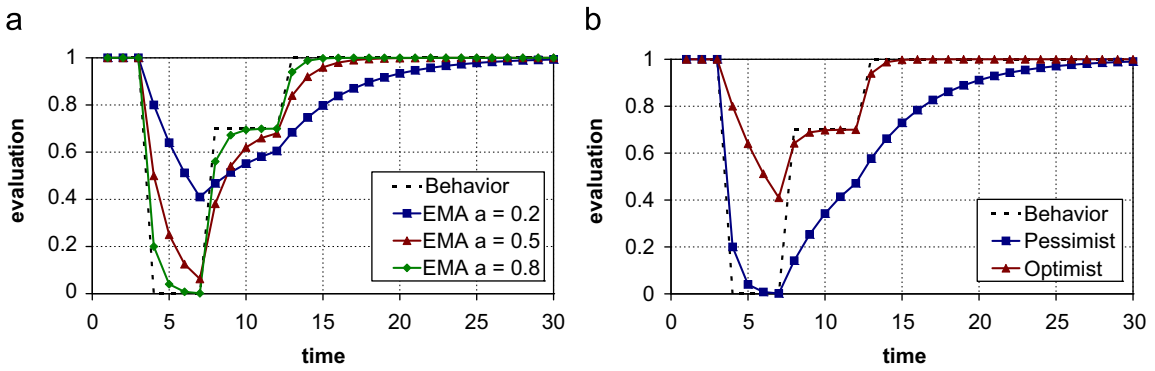

Fig. 6. Smoothing of trust values over time: (a) evolution of trust applying EMA and (b) optimistic and pessimistic perception of trust modeled with adaptive EMA.

behavior. If an actor once behaved untrustworthy, it will likely take some time to regain full trust again. Therefore, depending on the selected tuning parameter α, different strategies for merging current trust values with the history can be realized. According to Eq. (14), for $\alpha > 0.5$ the actual behavior is counted more, otherwise the history gains more importance. Fig. 6(a) shows three smoothened time lines, calculated with different smoothing factors. There exist several other approaches to trust evolution which work with deep histories, e.g., [31], however, EMA requires less memory and lower computational effort.

As shown in Fig. 6(a), by applying EMA previous or current behavior is given more importance. However, personal traits, such as being optimistic or pessimistic, demands for more sophisticated rules of temporal evaluation. In our case, we define an optimist as somebody who predominantly remembers positive and contributing behavior and tends to quickly forgive short-term unreliability. In contrast to that, a pessimist loses trust also for short-term unreliability and needs more time to regain trust than the optimist. Examples of this behavior are depicted by Fig. 6(b). Optimistic and pessimistic perceptions are realized by adapting the smoothing factor α according to Eq. (15). Whenever the curve depicted in Fig. 6(b) changes its sign, τ_i^s is re-calculated with adapted α. A small deviation ε denotes that the smoothing factor is either near 0 or near 1, depending on falling or rising trustworthiness. An enhanced version of this approach may adapt parameters in more fine-grained intervals, for instance, by considering lower and higher drops/rises of trustworthiness.

$$\alpha = \begin{cases} 0+\varepsilon & \text{if } \textit{optimistic} \text{ and } \tau_i^s < \tau_{i-1}^s \\ 1-\varepsilon & \text{if } \textit{optimistic} \text{ and } \tau_i^s \geq \tau_{i-1}^s \\ 0+\varepsilon & \text{if } \textit{pessimistic} \text{ and } \tau_i^s \geq \tau_{i-1}^s \\ 1-\varepsilon & \text{if } \textit{pessimistic} \text{ and } \tau_i^s < \tau_{i-1}^s \end{cases} \quad (15)$$

6.3. Trust projection

The concepts of *trust projection* combine existing personal trust relations or compare the similarity of profiles to predict potentially emerging trust relations. Projected relations, predicted by the means of recommendation or reputation, are fundamentally different from personal trust relations, because they do not rely on personal experience, therefore, not personally proven.

Trust mapping: τ_{map} describes a mechanism to predict a trust relation in a particular scope s, when trust relations between the same actors in different, but to some extent similar scopes $S = \{s_x | \tau^{s_x}(u,v) \neq \perp\}$, already have been established. For instance, if a trust relation in an engineer regarding software implementation activities has already been established, this relation can be mapped to software modeling as well. However, a good programmer may not be a good software architect, the software programmer may be trusted because of his knowledge, experience, and expertise, that are similar for both types of activities. Therefore, we define trust mapping to rely on the similarity of scopes. Trust $\tau^{s_x}(u,v)$ can be mapped to scope s. For that purpose each τ^{s_x} is weighted *and* attenuated by the scope similarity, as shown in Eq. (16). We assume that the function $sim_{scope}(s_1,s_2)$ returns the similarity of two scopes between 0 (totally different) and 1 (totally equal). An example of sim_{scope} for tag-based scope definitions, is measuring the amount of matching tags:

$$\tau_{map}^s(u,v) = \frac{\sum_{s_x \in S} \tau_x^s(w,v) \cdot (sim_{scope}(s_x,s))^2}{\sum_{s_x \in S} sim_{scope}(s_x,s)} \quad (16)$$

Recommendation: $\tau_{rec}^s(u,v)$ is built by aggregating u's trustees' trust relations to v. Recommendation represents therefore second-hand experiences. Potential recommenders of u for v are all $Rec \subseteq \{n \in V | \tau^s(u,n) \neq \perp \wedge \tau^s(n,v) \neq \perp\}$. According to Fig. 7(c), the recommenders' $Rec = \{w,x\}$ perception of trust will likely be different from the actor's u perception (that is receiving the recommendation), because all of them define trust upon different rule sets. For instance, optimists generally tend to provide better recommendations of third parties than pessimists. Considering p_τ^s allows to account for differences in trust perceptions between the set of recommenders Rec and a user u of recommendations. Thus, u could define to utilize only recommendations of trustees having similar perceptions of trust, i.e., $p_\tau^s(u) \approx p_\tau^s(w) \approx p_\tau^s(x)$. As common in

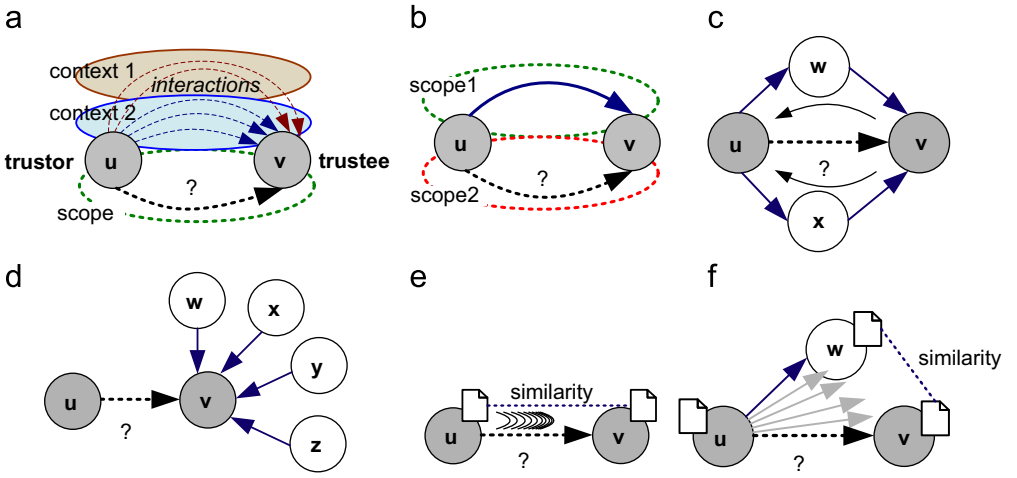

Fig. 7. Trust inference concepts: (a) direct trust inference; (b) trust mapping; (c) recommendation; (d) reputation; (e) trust mirroring and (f) trust teleportation.

other models, e.g., [7], we weight the recommendation of each $w \in Rec$ with the trustworthiness of u in w (Eq. (17)):

$$\tau^s_{rec}(u,v) = \frac{\sum_{w \in Rec} \tau^s(w,v) \cdot \tau^s(u,w)}{\sum_{w \in Rec} \tau^s(u,w)} \tag{17}$$

Reputation: τ_{rep} is similar to recommendation, however, according to Fig. 7(d) the actor u inferring the reputation of v does not require a personal trust relation to v's trustors $\{w,x,y,z\}$ ('reputing' entities $Rep \subseteq \{n \in V | \tau^s(u,n) \neq \perp\}$ described by a column vector of the matrix $\mathfrak{T}^s$). Reputation represents a kind of global (community) trust, calculated on top of each trustor's personal trust relations. Because these relations rest upon individually set-up rules, the results of our reputation model reflect someone's *real* standing, influenced by the subjective trust perceptions of his trustors. As in other models, single trust relations can be weighted before aggregation, e.g., according to the similarities of respective p^s_τ (Eq. (18)). More advanced models may account for the reputation of the trustors, leading to a PageRank-like model (see also TrustRank [32]):

$$\tau^s_{rep}(u,v) = \frac{\sum_{w \in Rep} \tau^s(w,v) \cdot sim_{percep}(p^s_\tau(u), p^s_\tau(w))}{\sum_{w \in Rep} sim_{percep}(p^s_\tau(u), p^s_\tau(w))} \tag{18}$$

Trust mirroring: τ_{mir} is typically applied in environments where actors have the same roles (e.g., online social platforms). Depending on the environment, interest and competency similarities of people can be interpreted directly as an indicator for future trust (Eq. (19)). There is strong evidence that actors 'similar minded' tend to trust each other more than any random actors [33,34]; e.g., movie recommendations of people with same interests are usually more trustworthy than the opinions of unknown persons. Mirrored trust relations are directed, iff $sim_{profile}(P(u),P(v)) \neq sim_{profile}(P(v),P(u))$. For instance an experienced actor v might have at least the same competencies as a novice u in a scope s (expressed by the profile P^s tailored for s). Therefore, v covers mostly all competencies of u and $\tau^s_{mir}(u,v)$ is high, while this is not true for $\tau^s_{mir}(v,u)$. If $sim_{profile}(P(u),P(v)) = sim_{profile}(P(v),P(u))$, then $\tau^s_{mir}(u,v)$ can be mirrored along the main diagonal of the trust matrix $\mathfrak{T}^s$ to $\tau^s_{mir}(v,u)$:

$$\tau^s_{mir}(u,v) = sim_{profile}(P^s(u), P^s(v)) \tag{19}$$

Trust teleportation: τ_{tel} is applied as depicted in Fig. 7(f). We assume that u has established a trust relationship to w in the past, for example, based on w's capabilities to assist u in work activities. Therefore, others having interests and capabilities similar to w may become similarly trusted by u in the future. In contrast to mirroring, trust teleportation is applied in environments comprising actors with different roles. For example, a manager might trust a software developer belonging to a certain group. Other members in the same group may benefit from the existing trust relationship by being recommended as trustworthy as well. We attempt to predict the amount of future trust from u to v by comparing w's and v's profiles P^s, especially the parts relevant in scope s. Eq. (20) deals with a generalized case where several trust relations from u to members of a group $Tele(\subset \vec{T}^s(u))$ are teleported to a still untrusted actor v. Similar to trust mapping, teleportation is weighted and attenuated by the similarity measurement results for actor profiles (see [12] for details on profile similarity measurement):

$$\tau^s_{tele}(u,v) = \frac{\sum_{w \in Tele} \tau^s_x(u,w) \cdot (sim_{profile}(P^s(w), P^s(v)))^2}{\sum_{w \in Tele} sim_{profile}(P^s(w), P^s(v))} \tag{20}$$

7. Towards flexible compositions

Established trust relations advise network members to keep interacting with successful (and thus trusted) collaboration partners, and to avoid—or even refuse—interactions with partners from unsuccessful collaborations. However, recommending actors to collaborate only with their most trusted partners might have negative side effects for the whole community. For instance in our Expert Web scenario, some actors will provide support

highly above average (similar to common Internet forums). Therefore, as a small amount of network members become very popular, they will also become more and more overloaded with help and support requests. Furthermore, once somebody lost his trustors, it will be hard—if not impossible—for him/her to regain trust. An actor might lose his/her trustors not only if s/he behaves untrustworthy, but also if another one behaves more trustworthy (with respect to the configured trust inference rules). We introduce two concepts for compensating this load balancing problem that is mostly neglected, but of paramount importance in collaborative environments:

- *Community balancing models* enforce network members to select interaction partners not only based on trust, but considering at least one further impact, such as costs (trade-off).
- *Request delegation patterns* enable network members to delegate incoming request to third-parties, in case they are overloaded. On the one side, this method balances the work load within a community, on the other side, it enables actors to establish trust relations to still unknown members.

7.1. Community balancing models

Balancing communities is essential to distribute workload in professional environments, and to facilitate the participation of erratically involved actors and newcomers in social communities. For instance, in the Expert Web use case of this paper, the load should be balanced between all members of the network. This means on the one side highly reputed actors should not become flooded with requests (e.g., for each help request, independent from how important or challenging it is, only actors with highest reputation would be chosen). On the other side, newcomers and lower reputed actors should get a chance to increase their standing and visibility within the community. Therefore, mechanisms are needed to let actors not only select experts based on trust but also further influencing factors. That is realized with *trust trade-off models*.

The applicability of trade-off models is highly dependent from the domain and environment. For example, for a non-profit social Web platform, where people request suggestions for planning their holidays, a trade-off model will be fundamentally different from a business oriented help and support network. Therefore, we categorize balancing models in the following two classes:

- *Business-oriented models* rely on mechanisms of the free economy, including supply and demand. For instance, highly demanded services (usually from top reputed actors) cause higher costs on the consumer side than services offered by ordinary actors. Concepts to realize rewarding of services and payment for contributions are needed. In the Expert Web use case of this paper, actors could earn rewards for providing reliable and successful support, and would have to pay for involving and interacting with other experts.
- *Social-oriented models*, as typically applied in free and open social platforms, have other concepts for balancing communities. For instance, reciprocity [9] reflects the amount of obtained benefit from the network compared to provided contribution. Actors may be allowed to interact with high reputed community members, if they do not only exploit the network, but also contribute to a certain extent.

For both types of balancing models, reliable interaction behavior and expedient support of community members is rewarded. In a large-scale SOA environment, where potentially thousands of actors may interact and are flexibly composed for short-term collaborations, enabling automatic rewarding is highly beneficial. Instead of rating someone's contribution manually (typical human feedback), we attempt to apply automatic rewarding as far as suitable. For instance, tracking, if sent requests trigger responses, and average time spans for processing requests are common quality of service (QoS) measures in service-oriented systems. However, simple structural interaction analysis does not account for the value of interactions. Therefore, on a more abstract level, contributions can be, at least to some extent, automatically tracked. For instance, achieving a predefined activity goal (milestone), such as creating a project artifact, or starting a follow-up activity in a process-centric environment, are strong indicators that preceding collaborations have been successful.

7.1.1. Business-oriented models

These models comprise concepts from the free economy, such as *supply* and *demand*, and *costs* and *risks*. In business-oriented models as applied for instance in the motivating scenario of this paper, each actor, offering support and services to others, can demand a dynamically adjustable amount of credits from service consumers. The amount of this compensation is usually directly proportional to the demand of the service, equally to the free economy. Usually high quality services with higher reputation are demanded more by network members. On the one side, this mechanism motivates actors to behave trustworthy (because than they will earn higher rewards), on the other side paying for services will urge consumers to distribute their requests between expensive (e.g., highly reputed) and cheaper (e.g., newcomer) services in the network.

7.1.2. Social-oriented models

In contrast to business-oriented system, socially inspired environments do not (primarily) account for payment. Normally, the participation in such networks is for free, however, members expect others to participate in a beneficial way. For instance, in a sharing portal, actors share files for free, however, they expect the same from others. This concept is described by *reciprocity*. This definition of reciprocity, motivated by studies in which repeated games are played between individuals [35],

describes the expectation that actors will respond to each other in similar ways, i.e., with similar benevolence of their own. More general, reciprocity describes an actor's contributing behavior with respect to the whole community (social reciprocity). This concept can be applied in a wide range of social networks, to enable a kind of 'soft access control system'.

In a non-commercial version of the Expert Web, such an access control system can evaluate for each actor which network members (providing support services) can be contacted and interacted with. For that purpose, we define reciprocity as the ratio of obtained help from the network (i.e., delegated tasks to others), and provided help (i.e., accepted and processed tasks from others).

7.2. Request delegation patterns

A common problem of trust and reputation mechanisms in online communities is that there emerge only a minority of highly trusted actors, while the majority remains in the background. Therefore, network members tend to consult and interact with the same (already trusted) partners again and again, leading to work overloads of these actors, and hindering the emergence of new trust relations. We utilize the means of delegations to compensate this load and interaction balancing problem that is often neglected, but of paramount importance in collaborative environments. In the motivating Expert Web scenario of this paper, actors send and process requests for support (RFS). Once an actor gets overloaded s/he should be able to delegate requests to other actors (with potentially free resources). If the receivers of such delegations behave trustworthy, i.e., respond fast and reliably, the original requesters will establish trust to them. Fig. 8 visualizes this model. In case of a successful delegation, u sends an RFS to v who delegates to w, and w responds directly to u. This interaction will positively impact the metrics that describe the relation from u to w, and finally $\tau(u,w)$ increases. The relation $\tau(u,v)$ is neither rewarded nor punished, because on the one side v did not serve u's RFS, but on the other side, v was able to successfully delegate, and thus did not harm u. The relation $\tau(v,w)$ is also not influenced, since v is not the original requester. If a delegation fails (Fig. 8(b)), i.e., an RFS is not responded, metrics that describe both $\tau(u,v)$ and $\tau(v,w)$ are negatively influenced (for instance the success rate is decreased), because of v's and w's unreliable behavior. But in that case, we assume that $\tau(u,w)$ remains unchanged. Although w has not served u's request, we do not know the reasons for that behavior. For instance, a denial of service attack could maliciously harm w's reputation (the sum of trust relations), if s/he is flooded with delegated RFSs.

The described delegation mechanisms and their influence on trust are configured by a domain expert in VieTE, and are feasible for our Expert Web scenario, where all participants in the network have similar collaboration roles (in particular to provide help and support). Other delegation and trust mechanisms, accounting for different roles of network members and restrictions of delegations due to confidentiality reasons, may be desirable in other domains.

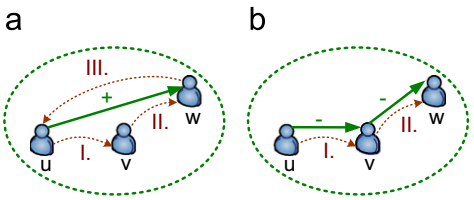

Fig. 8. Delegations and their impact on trust: (a) successful delegation and (b) failed delegation.

One of the major challenges to enable sophisticated balancing is to determine the receivers of delegations in the whole network. Usually, the selection will rely on trust, because, as shown in Fig. 8(b), it is in v's interest to delegate successfully and not to get punished. A fundamental selection strategy randomly picks an actor from a pool of service providers that are personally trusted above a pre-defined limit. Based on each individual's interaction history, every network member has his/her own pool of trusted actors. More advanced selection models are out of scope of this paper, and are subject to further research.

8. Architecture and implementation

We discuss the *VieTE—Vienna Trust Emergence Framework* [5,36] to research and evaluate novel concepts of trust and reputation in mixed systems environments. An overview of the framework is provided in Fig. 9. Briefly, the system captures various kinds of interactions, calculates metrics, such as responsiveness, skillLevel, availability, performs a personalized rule-based interpretation of these metrics, and finally infers trust between each pair of interacting members. The main components, developed as SOAP-based Web services in Java, hosted by Axis2,[5] are as follows:

Interaction monitoring: Interactions are either captured by interaction sensors, included in infrastructure services, or external access layers. An example for the first case is the activity management service that notifies a logging service about activity delegations and assignments. In the second case, service invocations via SOAP are routed over an access layer that captures the SOAP messages. Later on, VieTE periodically analyzes captured interactions offline and calculates higher level interaction metrics. While the depicted architecture follows a centralized approach, the logging facilities are replicated for scalability reasons, and monitoring takes place in a distributed form. Interactions are purged in predefined time intervals, depending on the required depth of history needed by metric calculation plugins.

Activity management: Actors use activities to manage their work as introduced before. Activities are structures to describe work and its goals, as well as participating actors, used resources, and produced project artifacts.

[5] http://ws.apache.org/axis2/

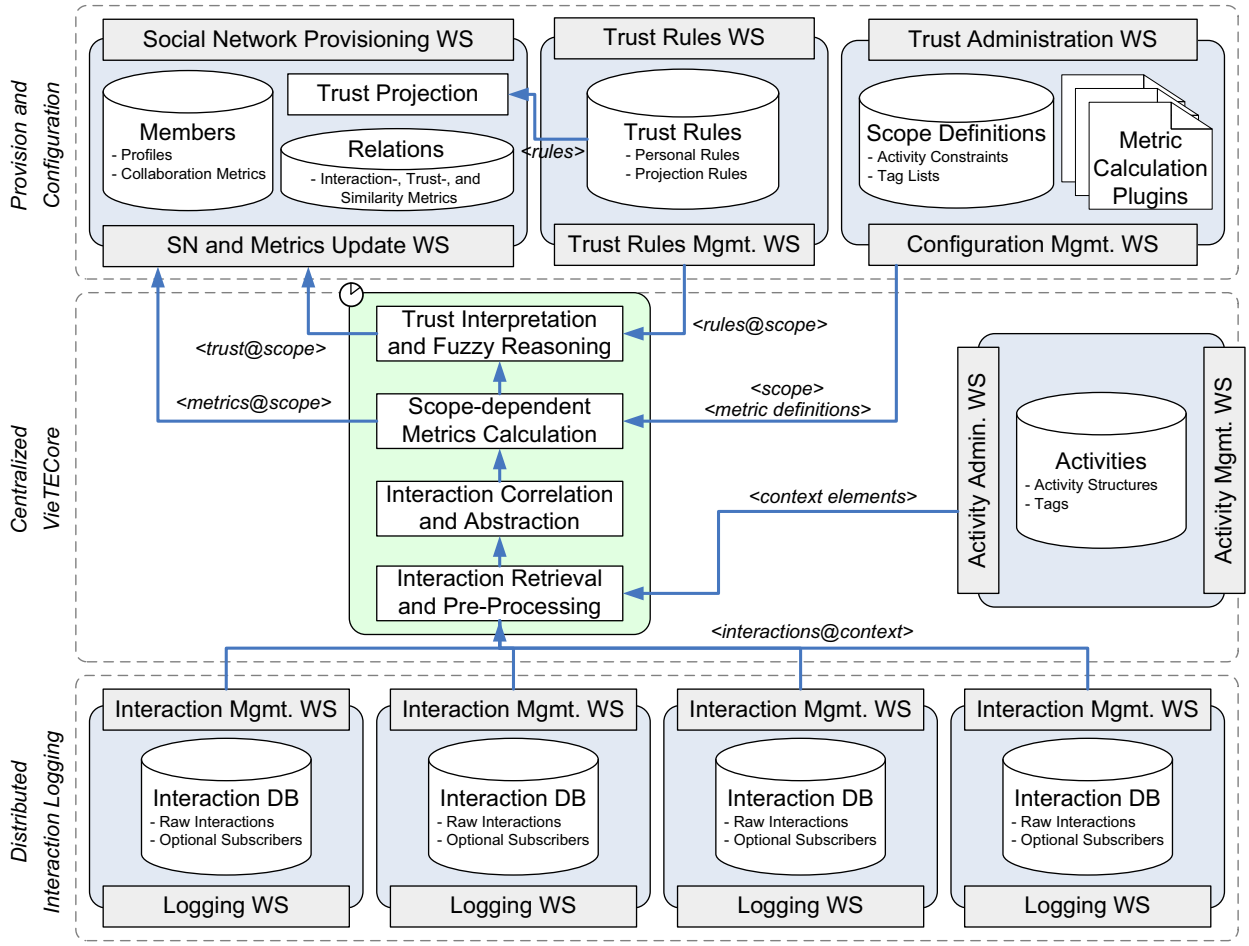

Fig. 9. VieTE framework overview.

A detailed description of this model, used to capture the context of interactions, is provided by [5].

Trust model administration: A domain expert configures certain properties of the trust inference process that are applied for all participants of the network. For instance, s/he defines meaningful trust scopes in the given domain and business area, configures available metric calculation plugins that provide the metrics for personal trust rules, and sets up the general trust model behavior, such as temporal constraints for interaction analysis and end-points of logging facilities.

Personal trust rules management: Personal trust rules, configured by each member of the network, enable the inference of a subjective and personalized view on trust. On the one side a fuzzy inference engine utilizes sets of trust rules to interpret available interaction metrics and establish personal trust in predefined scopes. On the other side rules configure the mode of operation of some higher level *trust projection* concepts to establish relations between non-interacting network members.

Social network management and provisioning: This component enables the registration of humans and services, including their individual profiles. All registered actors build the set of vertices of a trust graph $G=(V, E)$ (see the *Web of Trust* in [3]). We support the discovery of actors during ongoing collaborations (similar to a Web service registry), relying on actor capabilities (profiles), and periodically inferred metrics (interaction-, similarity-, collaboration-, and trust metrics). Furthermore, for evaluating the trustworthiness of small sets of actors, flexibly applied concepts of trust projection can be used. An example use case is the composition of actors to a *human–service ensemble*, where a group of potential participants is pre-selected based on their capabilities or professional affiliations, and finally a subset of them flexibly composed with respect to their trust relations. This mechanism enables late binding, as common in a (Web) services world.

VieTECore: The VieTE core is the heart of the VieTE framework and connects all the aforementioned components. As depicted in Fig. 9 it retrieves interactions from the replicated interaction management services, performs an analysis and calculates interaction metrics using metric plugins. Finally, updated metrics are interpreted and personal trust is inferred utilizing a fuzzy rule engine. These actions are periodically scheduled, e.g. on a daily or weekly basis.

8.1. Human provided services in the expert web

An excerpt of the RFS schema definitions is shown in Listing 2 defining complex data structures.

Listing 2. RFS schema definition.

```
<xsd:schema tns="http://myhps.org/rfs">
 <xsd:complexType name="GenericResource">
  <xsd:sequence>
   <xsd:element name="Location" type="xsd:anyURI"/>
   <xsd:element name="Expires" type="xsd:dateTime"/>
  <xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="Request">
  <xsd:sequence>
    <xsd:element name="subject" type="xsd:string"/>
    <xsd:element name="requ" type="xsd:string"/>
    <xsd:element name="resource" type="GenericResource"/>
    <xsd:element name="keywords" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="SupportRequest" type="Request"/>
 <xsd:element name="AckSupportRequest" type="xsd:string"/>
 <xsd:element name="GetSupportReply" type="xsd:string"/>
 <!-- reply details omitted -->
 <xsd:element name="SupportReply" type="Reply"/>
</xsd:schema>
```

Listing 3 shows the binding of the HPS WSDL to the (HPS) infrastructure services.

Listing 3. HPS WSDL binding.

```
<!-- excerpt wsdl interface -->
<wsdl:portType name="HPSSupportPortType">
 <wsdl:operation name="GetSupport">
  <wsdl:input xmlns="http://www.w3.org/2006/05/addressing/wsdl"
   message="GetSupport" wsaw:Action="urn:GetSupport">
  </wsdl:input>
  <wsdl:output message="AckSupportRequest" />
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="HALSOAPBinding" type="HPSSupportPortType">
 <soap:binding style="document"
  transport="http://xmlsoap.org/soap/http"/>
</wsdl:binding>
```

The `GenericResource` defines common attributes and metadata associated with resources such as documents or policies. A `GenericResource` can encapsulate remote resources that are hosted by a collaboration infrastructure (e.g., document management). `Request` defines the structure of an RFS (here we show a simplified example). A `Reply` is the corresponding RFS response (we omitted the actual XML definition). The protocol (at the technical HPS middleware level) is asynchronous allowing RFSs to be stored, retrieved, and processed. For that purpose we implemented a middleware service (HPS Access Layer—HAL) which dispatches and routes RFSs. `GetSupport` depicts a WSDL message corresponding to the RFS `SupportRequest`.

Upon receiving such a request, HAL generates a session identifier contained in the output message `AckSupportRequest`. A notification is sent to the requester (assuming a callback destination or notification endpoint has been provided) to deliver RFS status updates for example; processed RFSs can be retrieved via `GetSupportReply`. The detailed notification mechanism can be found in [23].

8.2. Interaction monitoring and logging

The HPS Access Layer logs each service interaction (request and response message) through a logging service. RFSs and their responses, exchanged between community members, are modeled as traditional SOAP calls, but with various header extensions, as shown in Listing 4.

Listing 4. Simplified RFS via SOAP example.

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:vietypes="http://viete.infosys.tuwien.ac.at/Type"
 xmlns:hps="http://myhps.org/"
 xmlns:rfs="http://myhps.org/rfs">
 <soap:Header>
  <vietypes:timestamp value="2009-03-05"/>
  <vietypes:delegation hops="3" deadline="2009-03-06"/>
  <vietypes:activity url="http://www.coin-ip.eu/Activity#42"/>
  <wsa:MessageID>uuid</wsa:MessageID>
  <wsa:ReplyTo>http://.../Actor#Florian</wsa:ReplyTo>
  <wsa:From>http://.../Actor#Florian</wsa:From>
  <wsa:To>http://.../Actor#Daniel</wsa:To>
  <wsa:Action>http://.../Type/RFS</wsa:Action>
 </soap:Header>
 <soap:Body>
  <hps:Request>
   <rfs:subject>WSDL consumption with Axis2?</rfs:subject>
   <rfs:requ>Axis2 reports a parsing error while consuming
             the given resource. What is wrong?</rfs:requ>
   <rfs:resource>
      <!-- details omitted -->
   </rfs:resource>
   <rfs:keywords>WSDL, Axis2</rfs:keywords>
  </hps:Request>
 </soap:Body>
</soap:Envelope>
```

The most important extensions are:

- `Timestamp` capture the actual creation of the message and is used to calculate temporal interaction metrics, such as average response times.
- `Delegation` holds parameters that influence delegation behavior, such as the number of subsequent delegations `numHops` (to avoid circulating RFSs) and hard `deadlines`.
- `Activity uri` describes the context of interactions (see [37] for activity model).
- `MessageID` enables message correlation, i.e., to properly match requests and responses.
- `WS-Addressing` tags, besides MessageID, are used to route RFSs through the network.

8.3. Metric calculation

Metrics describe the interaction behavior and dynamically changing properties of actors. Currently, we account for the metrics described in Table 1 for trust interpretation upon logged SOAP calls in the Expert Web scenario. Note, as described before, these metrics are determined for particular scopes; i.e., based on a subset of interactions that meet certain constraints. The availability of a service, either provided by humans or implemented in Software, can be high in one scope, but much lower in another one. Furthermore, these metrics are calculated for each directed relation between pairs of network members. An actor u might serve v reliably, but not a third party w.

Table 1
Metrics utilized for trust inference.

Metric name	Range	Description
Availability	[0,100]	Ratio of accepted to received RFSs
Response time	[0,96]	Average response time in hours
Success rate	[0,100]	Amount of successfully served RFSs
Experience	$[0,\infty]$	Number of RFSs served
RFS reciprocity	$[-1,1]$	Ratio of processed to sent RFSs
Manual reward	[0,5]	Optional manually assigned scores
Costs	[0,5]	Price for serving RFSs

Our approach relies on mining of metrics, thus, values are not manually entered but are frequently updated by the system. This enables collaboration partners to keep track of the dynamics in highly flexible large-scale networks. Besides interaction behavior in terms of reliability or responsiveness, also context-aware experience mining can be conducted. This approach is explained in detail in [23].

In trust inference examples in previous sections, we accounted for the *average response time* t_r (Eq. (21)) of a service and its *success rate sr* (Eq. (22)). These are typical metrics for an *emergency help and support environment*, where fast and reliable support is absolutely required, but costs can be neglected. We assume, similar complexity of requests for support (RFS) in a scope *s*, thus different RFSs require comparable efforts from services (similar to a traditional Internet forum).

The response time is calculated as the duration between sending (or delegating) a request (t_{send}) to a service and receiving the corresponding response ($t_{receive}$), averaged over all served RFSs. Unique IDs of calls (see SOAP header in Listing 4) enable sophisticated message correlation to identify corresponding messages:

$$t_r^s = \frac{\sum_{rfs \in RFS}(t_{receive}(rfs) - t_{send}(rfs))}{|RFS|} \tag{21}$$

An RFS is considered successfully served (*sRFS*) if leading to a result before a predefined deadline, otherwise it fails (*fRFS*):

$$sr^s = \frac{num(sRFS)}{num(sRFS) + num(fRFS)} \tag{22}$$

8.4. Trust provisioning

The `Social Network Provisioning WS` (see Fig. 9) is a WSDL-based Web Service that provides the dynamically changing *Web of Trust* as standardized directed graph model. It is a major part of the VieTE framework and used by other services, such as partner discovery tools, to retrieve social relations for service personalization and customization in virtual communities. The Web service interface deals with the following fundamental types of entities:

- `Vertex`: A vertex describes either a human, software service, or HPS.
- `Edge`: An Edge reflects the directed relation between two vertices.
- `Metric`: Metrics describe properties of either vertices (such as the number of interactions with all partners, or the number of involved activities) or edges (such as the number of invocations from a particular service by a particular human). Metrics are calculated from interactions and provided profiles with respect to pre-configured rule sets (e.g., only interactions of a particular type are considered in the trust determination process).
- `Scope`: Rules determine which interactions and collaboration metrics are used for trust calculation. These rules describe the constraints for the validity of calculated metrics, i.e., the scope of their application. Common scopes are pre-configured and can be selected via the Web Service interface.

The Social Network Provisioning WS enables the successive retrieval of the Web of Trust starting with a predefined vertex, e.g., reflecting the current service user. We specify its interface as shown in Table 2. Note, for data retrieval, metrics are merged in the entities vertex and edge. All entities are identified by an URI, which is a combination of a

Table 2
Social network provisioning WS interface specification.

Method name	Parameter	Description
`getVertex`	vertexURI	Get the vertex object with the given uri
`getVerticesByName`	vertexName (regex)	Get a list of vertices with matching names
`getAllVertices`	–	Get all vertices (can be restricted to a maximum number due to performance reasons)
`getEdge`	edgeURI	Get the specified edge
`getEdges`	sourceVertexURI, sinkVertexURI	Get all directed edges from sourceVertex to sinkVertex
`getOutEdges`	sourceVertexURI	Get all out edges of the specified vertex
`getInEdges`	sinkVertexURI	Get all in edges of the specified vertex
`getScope`	scopeURI	Get one particular scope in the network
`getAllScopes`		Get all available scopes in the network
`getSourceVertex`	edgeURI	Get the vertex object which is the source of the given edge
`getSinkVertex`	edgeURI	Get the vertex object which is the sink of the given edge
`getNeighbours`	vertexURI, numHops	Get neighbors (independent of edge orientation); the optional parameter numHops may set the maximum path length from the specified vertex to resulting neighbours
`getSuccessors`	sourceVertexURI	Get successors of specified vertex
`getPredecessors`	sinkVertexURI	Get direct predecessors of specified vertex
`getVersion`	–	Get version string

`basepath` (e.g., http://www.infosys.tuwien.ac.at/coin), the `entity type` (e.g., vertex) and an integer `id`.

9. Evaluation and discussion

In this section, we show the *results of performance evaluations* that discuss major design decisions and VieTE's applicability in large-scale networks; and a *functional evaluation* that deals with the actual application of our trust inference approach for balancing communities.

9.1. Computational complexity of trust management

A fundamental aspect of our trust management approach is the context-awareness of data and social relations. Due to the high complexity of large-scale networks comprising various kinds of interactions and distinct scopes of trust, we evaluate the feasibility of our framework by well-directed performance studies. We focus on the most critical parts, i.e., potential bottlenecks, in our system, in particular, on (i) trust inference upon interaction logs, (ii) profile similarity measurement for trust mirroring and teleportation, (iii) the calculation of recommendations based on mined graph structures and (iv) provisioning of graph segments to users. The conducted experiments address general technical and research problems in complex networks, such as emerging relations in evolving structures, graph operations on large-scale networks, and information processing with respect to contextual constraints.

9.1.1. Experiments setup and data generation

For conducting our performance studies, we generate an artificial interaction and trust network that we would expect to emerge under realistic conditions. For that purpose we utilize the *preferential attachment model* of Barabasi and Albert to create[6] network structures that are characteristic for *science collaborations* [38]. As shown in Fig. 10 for a graph with 500 vertices, the output is a scale-free network with vertex degrees[7] following a power-law distribution. These structures are the basis for creating realistic interaction logs that are used to conduct trust inference experiments. For a graph G, we generate in total $100 \cdot |E|$ interactions between pairs of vertices (u,v). In our experiments, we assume that 80% of interactions take place between 20% of the most active users (reflected by hub vertices 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. While we payed attention on creating a realistic amount and distribution of interactions that are closely bound to vertex degrees, the interaction properties themselves, i.e., type, timestamp, do not influence the actual performance study (because they do not influence the number of required operations to process the interaction logs).

For the following experiments, VieTE's trust provisioning service is hosted on a server with Intel Xeon 3.2 GHz (quad), 10 GB RAM, running Tomcat 6 with Axis2 1.4.1 on Ubuntu Linux, and MySQL 5.0 databases. The client simulation that retrieves elements from the managed trust graph runs on a Pentium 4 with 2 GB on Windows XP, and is connected with the server through a local 100 MBit Ethernet.

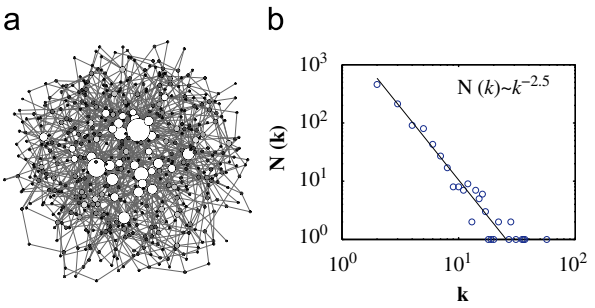

Fig. 10. Generated network applying preferential attachment: (a) scale-free graph structure and (b) power-law distribution.

9.1.2. Trust inference performance

Through utilizing available interaction properties, we calculate the previously discussed metrics (i) *average response time* t_r, and (ii) *success rate* sr (ratio of success to the sum of success and failure notifications). Individual response times are normalized to [0,1] with respect to the highest and lowest values in the whole network. The rule base to infer confidence between each pair of connected vertices has been shown in Listing 1. If the amount of interactions $|I(u,v)|$ between a pair (u,v) is below 10, we set the reliability of confidence to $|I(u,v)|/10$, else we assume a reliability of 1. Trust is calculated by multiplying confidence with its reliability.

Interactions take place in context of activities. Instead creating artificial activity structures, we randomly assign context elements to synthetic interactions. These elements are represented by tags that are randomly selected from a predefined list. This list holds 5 different tags, and each interaction gets 2–4 of them assigned. Such tags may describe the activity type where an interaction takes place, e.g., 'software development'; but also certain constraints, e.g., 'high risk'. We define 5 scopes, each described by exactly one possible tag. Thus, each interaction belongs to 2–4 scopes; and scopes may overlap. Interactions are uniformly distributed among scopes.

We measure the required time to completely process the synthetic interaction logs, including reading logs from the interaction database (SQL), aggregating logs and calculating metrics, normalizing metrics (here only the response time, because the values of the success rate are already in [0,1]), inferring trust upon a predefined rule base, and updating the trust graph (EMA with $\alpha = 0.25$). Experiments are performed for three networks of different sizes: small-scale with 100 vertices and 200 trust edges; medium-scale with 1000 vertices and 2000 edges; and large-scale with 10 000 vertices and 20 000 edges. Furthermore, trust is inferred (i) neglecting scopes (i.e., tags), (ii) for the defined scopes as above. The results in Table 3 show that especially for medium and large

[6] See JUNG graph library: http://jung.sourceforge.net.

[7] The vertex size is proportional to the degree; white vertices represent 'hubs'.

Table 3
Trust inference performance results.

Network characteristics	Mode	Computation time
Small-scale	No scopes	1 m 11 s
	5 scopes	1 m 56 s
Medium-scale	No scopes	11 m 41 s
	5 scopes	19 m 48 s
Large-scale	No scopes	109 m 03 s
	5 scopes	182 m 37 s

networks only a periodic offline calculation is feasible. Note, the difference of computational efforts accounting for no context (no scopes) and all scopes is not as high as one might expect. The reason is that a significant amount of time is required for SOAP communication in both cases.

9.1.3. Profile similarity measurement

Trust mirroring and *trust teleportation*, as explained in Section 6, rely on mechanisms that measure the similarities of actors in terms of skills, capabilities, expertise and interests. In contrast to common top-down approaches that apply taxonomies and ontologies to define certain skills and expertise areas, we follow a mining approach that addresses inherent dynamics of flexible collaboration environments. In particular, skills and expertise as well as interests change over time, but are rarely updated if they are managed manually in a registry. Hence, we determine and update them automatically through mining. However, since trust mirroring and teleportation are mainly used in the absence of interaction data, we need to acquire other data sources.

The creation of interest profiles without explicit user input has been discussed in [12]. That work assumes that users tag resources, such as bookmarks, pictures, videos, articles; and thus express their distinct interests. In particular, a dataset from `citeulike`[8] expresses people's use and understanding of scientific articles through individually assigned tags.

We use these data to create dynamically adapting interest profiles based on tags (ATPs—actor tagging profiles) and manage them in a vector space model [12]. However, since arbitrary tags may be freely assigned—there is no agreed taxonomy—no strict comparison can be performed. Therefore, we cluster tags according to their similarities and compare the actors' usage of tags on higher cluster levels. For instance, actors using tags belonging to the same cluster have similar interests, even if they do not use exactly the same tags. Hierarchical clustering enables us to regulate the fuzziness of similarity measurements, i.e., the size of tag clusters. The concrete mechanisms and algorithms are described in [12] and therefore out of scope of this work. But we outline the evaluation results of [12] to demonstrate the applicability of *automatic actor profile creation* and *cluster similarity measurement*, supporting the realization of trust mirroring and teleportation.

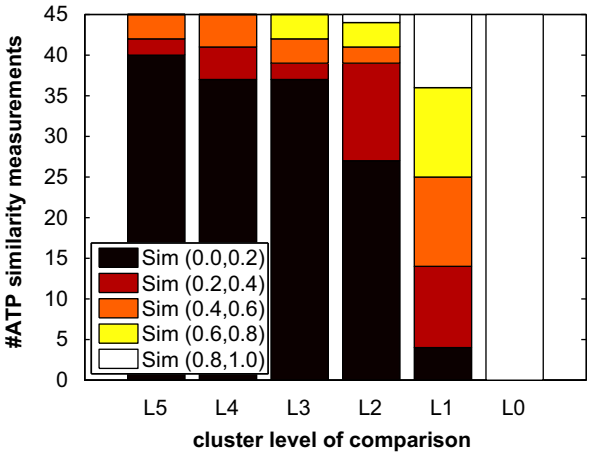

Fig. 11. Similarity results among 10 realistic actor tagging profiles (ATPs).

We determine for 10 representative `citeulike` users their tagging profiles (ATPs) in the domain of social networks. Then we compare these ATPs to find out to which degree actors use similar/same tags. The fundamental question is, if we are able to effectively distinguish similarities of different degrees among ATPs. In other words, in order to apply trust mirroring and teleportation we need distinguishable similarity results; and not e.g., all ATPs somehow similar. Fig. 11 shows the results of various profile similarity measurements. As explained, we compare profiles with varying fuzziness, i.e., on 5 different tag cluster levels. While on L5 each tag is in its own cluster, these clusters are consecutively merged until all tags are in the same cluster (L0). Hence, on L5 the most fine-grained comparison is performed, while on L0 all profiles are virtually identical. As shown, on L2 and L3 a small set of highly similar ATPs are identified, while the majority is still recognized as different. This is the desired effect required to mirror/teleport trust only to a small subset of available actors.

From a performance perspective, retrieving tags, aggregating and clustering them, and creating profiles takes some time. Especially mining these data on the Web is time-intensive. The overall performance highly depends on external systems that provide required data, such as `citeulike` in our case. Therefore, further performance studies have been omitted here.

9.1.4. Network management

This set of experiments, deal with managing trust in a graph model and the calculation of recommendation and reputation on top of a large-scale trust network with 10 000 vertices. Table 4 depicts the required time in seconds to calculate the recommendation $\tau^{s}_{rec}(u,v)$, having 10 and 100 recommender in the same scope (i.e., intermediate vertices on connecting parallel paths (u,v) of length 2).

[8] http://www.citeulike.org/

Table 4
Calculation times for τ^{s}_{rec} with 10 and 100 recommender.

Recommendation calculation method	10 rec. (s)	100 rec. (s)
Client-side	1.1	6.3
Servicer-side (SQL)	0.46	2.2
Server-side (in memory model)	0.28	0.34
Server-side (pre-calculation)	0.18	0.18

Several ways to implement recommendations exist. First, a client may request all recommender vertices and their relations and calculate recommendations on the client-side. However this method is simple to implement on the provider side, it is obviously the slowest one due to large amounts of transferred data. Still retrieving all recommender and relations directly from the backend database, but performing the calculation server-side, dramatically improves the performance. However, this method produces heavy load at the provider and its database and deems not to be scalable. Therefore, we map the network data, i.e., a directed graph model with annotated vertices and edges, in memory and perform operations without the backend database. Since all data are held in memory, the performance of calculating recommendations online is comparable to provisioning of pre-calculated data only. Hence, we design our system with an in-memory graph model, and further measure some aspects of this design decision. Fig. 12(a) illustrates required time for mapping the whole graph from the backend database to its in-memory representation. The effort increases linear with the number of vertices in the graph. Fig. 12(b) shows the memory consumption for graph instances of different sizes, first for the whole `Social Network Provisioning Service`, and second only for the graph object itself.

9.1.5. Trust graph provisioning

Retrieving trust values of certain relations, and even recommendations as shown before, causes minor computational effort. However, imagine someone frequently wants to calculate reputation based on network structures (see TrustRank [32]), would like to get notified if his neighborhood in the Web of Trust has grown to a certain size or if his collaboration partners have reached a particular experience level. Then, periodically retrieving larger segments of the trust graph G from the `Social Network Provisioning Service` is required. Therefore, we run some experiments to estimate the produced load in such situations.

The first experiment investigates the average size of potential collaboration partners who are either personally trusted or can be recommended (i.e., are connected through exactly one intermediate vertex). Experiment are conducted for various network sizes n and different average connection degrees of vertices. We pick random vertices from this set and run experiments for each of them until we calculate stable average results. Fig. 13 shows that in higher cross-linked networks (i.e., $\#trustees > 2$), personal relations and recommendations

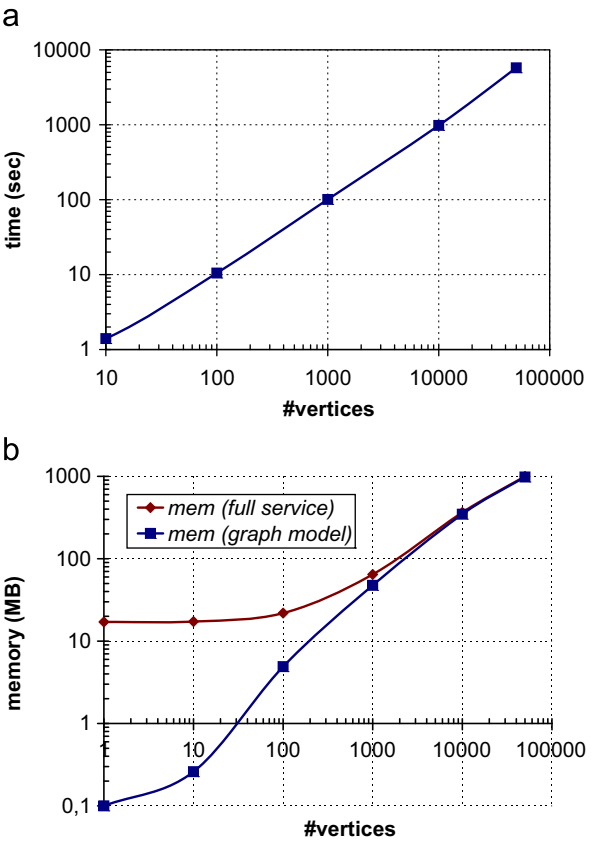

Fig. 12. Performance tests for mapping the graph model: (a) graph mapping time and (b) memory consumption.

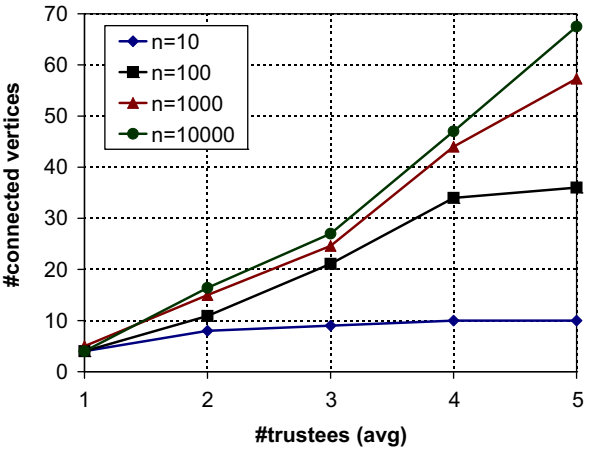

Fig. 13. Number of discovered potential collaboration partners through personal relations and recommendations for different network structures.

(so called 'second hand experiences') deem to be sufficient to discover new collaboration partners. However, in case of sparsely connected graphs, other mechanisms, such as trust mirroring or teleportation may be of high benefit.

Propagating trust over more than one intermediate vertex is of course possible (and widely applied), but leads

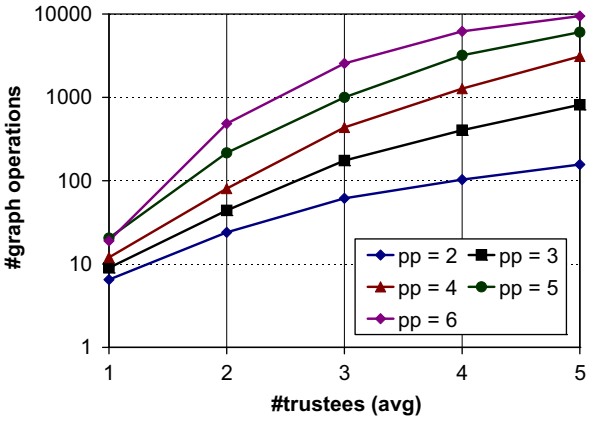

Fig. 14. Average number of required graph operations (for different average number of trustees) to determine all neighbors of a given vertex that are reachable on a path not longer than *pp*.

to significantly higher computational effort. Fig. 14 depicts the number of required graph operations depending on the average number of trustees (average outdegree of vertices). These graph operations mainly consist of retrieving vertices and edges, including their assigned metrics and trust values. For higher propagation path lengths *pp*, costs increase exponentially.

9.2. Interaction balancing in large-scale networks

We evaluate the functional application of the VieTE framework, by simulating typical scenarios in large-scale communities. In this experiment, we utilize the popular Repast Simphony[9] toolkit, a software bundle that enables round-based agent simulation. In contrast to researchers in the agent domain, we do not simulate our concepts by implementing different actor types and their behavior only, but we use a network of actors to provide stimuli for the actual VieTE framework. Therefore, we are not only able to evaluate the effectiveness of our new approach of fuzzy trust inference, but also the efficiency of the technical grounding based on Web service standards.

We focus on the motivational Expert Web use case from Section 2. In this scenario, a small set of simulated network members interact (sending, responding, and delegating RFSs), and these interactions are provided to the logging facilities of VieTE. The framework infers trust by calculating the described metrics t_r and *sr*, and using the rule set of Listing 1 for behavioral interpretation. Finally, emerging trust relations between the simulated actors influence the selection of receivers of RFSs. Hence, VieTE and the simulated actor network relies on each other, and are used in a cyclic approach; exactly the same way VieTE would be used by a real Expert Web. For this demonstration, all interactions take place in the same scope.

[9] http://repast.sourceforge.net

9.2.1. Simulation setup

Simulated agent network: Repast Simphony offers convenient support to model different actor behavior. As an inherent part of our environment, we make no distinction between human users and software services. Each actor owns a unique id (a number), creates SOAP requests, and follows one of the following behavior models: (i) *malicious actors* accept all RFSs but never delegate or respond, (ii) *erratic actors* accept all RFSs but only process (respond directly or delegate) RFSs originally coming from requesters with odd-numbered IDs, (iii) *fair players* process all requests if they are not overloaded, and delegate to trustworthy network neighbors otherwise.

We set up a network comprising 15 actors, where only one is highly reputed and fully trusted by all others as depicted in Fig. 15(a). This is the typical starting point of a newly created community, where one actor invites others to join.

VieTE setup: After each simulation step (round) seven randomly picked actors send one RFS to its most trusted actor (in the beginning this will only be the highly reputed one who starts to delegate). Each actor's input queue has exactly 5 slots to buffer incoming RFSs. A request is always accepted and takes exactly one round to be served. An actor processes an RFS itself if it has a free slot in its input queue, otherwise incoming RFSs are delegated to randomly picked trusted ($\tau > 0.8$) neighbors in the network. Note, one actor does not delegate more than one RFS per round to the same neighbor, however, an actor may receive more than one RFS from different neighbors in the same round. Delegations require one additional simulation round. There is an upper limit of 15 rounds for an RFS to be served (deadline); otherwise it is considered failed. A request can be delegated only three times (but not back to the original requester) (*hops*) to avoid circulating RFSs. Because the simulation utilizes only

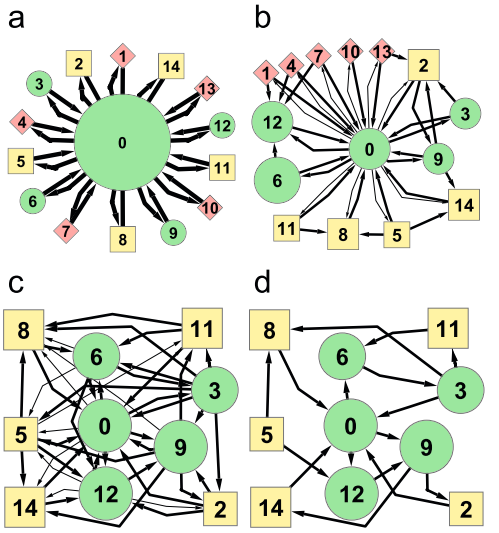

Fig. 15. Network structure after simulation round $n=\{0,100,250\}$. Elliptic vertices are fair players, rectangular shapes represent erratic actors, diamond shaped vertices reflect malicious actors: (a) initial $n=0$; (b) intermediate $n=100$; (c) balanced $n=250$ and (d) balanced (reduced).

two fully automatically determined metrics (t_r and sr), and no manual rewarding of responses, we assume an RFS is successfully served if a response arrives within 15 rounds (no fake or low quality responses). After each fifth round, VieTE determines t_r based on interactions in the most recent 25 rounds, and sr upon interactions in the last 50 rounds, and purges older logs. New values are merged with current ones using EMA with a fixed $\alpha = 0.25$.

9.2.2. Simulation results

We perform 250 simulation rounds of the described scenario with the aforementioned properties, and study the network structure in certain points of the simulation. The depicted networks in Fig. 15 show actors with different behavior and the temporal evolvement of trust relations between them. The size of the graph's vertices depend on the amount of trust established by network neighbors. Beginning with a star structure (Fig. 15(a)), the network structure in Fig. 15(b) emerges after 100 rounds, and Fig. 15(c) after 250 rounds, respectively. Note, since the behavior of actors is not deterministic (i.e., RFSs are sent to random neighbors that are trusted with $\tau > 0.8$ (lower bound of *full trust*; see Fig. 4)), the simulation output looks differently for each simulation run, however, the overall properties of the network are similar (number and strength of emerged trust relations).

In the beginning, all RFSs are sent to *actor 0*, who delegates to randomly picked trusted actors. If they respond reliably, requesters establish trust in that third parties. Otherwise they lose trust in *actor 0* (because of unsuccessful delegations). Therefore, actors with even-numbered IDs lose trust in *actor 0* faster than odd-numbered actors, because if *actor 0* delegates requests to erratic actors, they are not replied. As an additional feature in round 100, actors that are not trusted with $\tau > 0.2$ by at least on other network member, are removed from the network, similar to Web communities where *leechers* (actors that do not contribute to the network) are banned. Therefore, actors with malicious behavior disappear, while actors with erratic behavior still remain in the network. Fig. 15(d) shows a reduced view of the balanced network after 250 rounds. Only trust relations with $\tau > 0.8$ are visualized. As expected, most vertices have strong trust relations in at least one fair player (actors who reliably respond and delegate RFSs). However, remember that erratic actors reliably serve only requests coming from actors with odd-numbered IDs. Therefore, *actor 3* and *actor 9* also establish full trust in actors from this class. Note, if *actor 3* and *actor 9* would have re-delegated many RFSs coming from even-numbered actors to erratic actors, than those RFSs would have failed and only low trust would have emerged. However, due to the comparatively low load of the network (less than half of the actors receive RFSs per round (until n=100)), only a low amount of re-delegations occur (approx. 8% of RFSs).

10. Background and related work

Flexible and context-aware collaborations: In collaborations, activities are the means to capture the context in which human interactions take place. Activities describe the goal of a task, the participants, utilized resources, and temporal constraints. Studies regarding activities in various work settings are described in [39]. They identify patterns of complex business activities, which are then used to derive relationships and activity patterns [40,41]. The potential impact of activity-centric collaboration is highlighted [37] with special focus on the value to individuals, teams, and enterprises. Studies on distributed teams focus on human performance and interactions [42,43], even in Enterprise 2.0 environments [44]. Caramba [45] organizes work items of individuals as activities that can be used to manage collaborations. For example, one can see the status of an activity, who contributed to an activity, documents created within a particular activity, etc. Based on log analysis, human interaction patterns can be extracted [17].

Interactions in mixed systems: Major software vendors have been working on standards addressing the lack of human interaction support in service-oriented systems. WS-HumanTask [46] and Bpel4People [10] were released to address the emergent need for human interactions in business processes. These standards specify languages to model human interactions, the lifecycle of human tasks, and generic role models. Role-based access models [46] are used to model responsibilities and potential task assignees in processes. While Bpel4People based applications focus on top-down modeling of business processes, *mixed systems* target flexible interactions and compositions of Human-Provided and software-based services. This approach is aligned with the vision of the Web 2.0, where people can actively provide services. An example for a mixed system is a virtual organization (VO) using Web 2.0 technologies. A VO is a temporary alliance of organizations that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks [47]. Nowadays, virtual organizations are more and more realized with SOA concepts, regarding service discovery, service descriptions (WSDL), dynamic binding, and SOAP-based interactions. In such networks, humans may participate and provide services in a uniform way by using the HPS framework [2,23].

Behavioral and social trust models for SOA: Marsh [48] introduced trust as a computational concept, including a fundamental definition, a model and several related concepts impacting trust. Based on his work, various extended definitions and models have been developed. Some surveys on trust related to computer science have been performed [3,4,8], which outline common concepts of trust, clarify the terminology and describe the most popular models. From the many existing definitions of trust, those from [8,9] describe that trust relies on previous interactions and collaboration encounters, which fits best to our highly flexible environment. Context dependent trust was investigated by [3,4,8,48]. Context-aware computing focusing modeling and sensing of context can be found in [49,50].

Recently, trust in social environments and 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. A trust management framework for service-oriented environments has been presented in [51–53], however, without considering particular application scenarios with human actors in SOA. Although several models define trust on interactions and behavior, and account for reputation and recommendation, there is hardly any case study about the application of these models in service-oriented networks. While various theoretically sound models have been developed in the last years, fundamental research questions, such as the technical grounding in SOA and the complexity of trust-aware context-sensitive data management in large-scale networks are still widely unaddressed.

Depending on the environment, trust may rely on the outcome of previous interactions [1,9], *and* the similarity of interests and skills [11,12,33,34]. Note, trust is not simply a synonym for *quality of service* (QoS). Instead, metrics expressing social behavior and influences are used in certain contexts. For instance, *reciprocity* [9] is a concept describing that humans tend to establish a balance between provided support and obtained benefit from collaboration partners. The application of trust relations in team formations and virtual organizations has been studied before, e.g., in [54,55]. Trust propagation models [56–59] are intuitive methods to predict relations where no personal trust emerged; e.g., transitive recommendations.

In this paper, we described an approach to trust inference that is based on fuzzy set theories. This technique has been applied in trust models before [26–28], however, to our best knowledge, not to interpret diverse sets of interaction metrics. Utilizing interaction metrics, in particular calculated between pairs of network members, enables us to incorporate a personalized and social perspective. For instance, an actor's behavior may vary toward different network members. This aspect is usually out of scope in Web Services trust models, that are often closely connected to traditional QoS approaches [60].

Bootstrapping addresses the *cold start problem* and refers to putting a system into operation. Trust—from our perspective—cannot be negotiated or defined in advance. It rather emerges upon interactions and behavior of actors and thus, needs a certain time span to be built. However, until enough data has been collected, interests and skills can be used to predict potentially emerging trust relations. Mining, processing, and comparing user profiles is a key concept [11,12,33].

11. Conclusion and further work

Emerging service-oriented platforms no longer operate in closed enterprises. An increasing trend can be observed towards temporary alliances between companies requiring composition models to control and automate interactions between services. The resulting service-oriented application needs to be flexible supporting adaptive interactions. In this paper, we have motivated the need for adaptive interactions discussing an Expert Web scenario where people can register their skills and capabilities as services. Mixed service-oriented systems are *open* ecosystems comprising human- and software-based services. Trust mechanisms become essential in these systems because of changing actor interests and the dynamic discovery capabilities of SOA. Our trust model is based on fuzzy logic and rule-based interpretation of observed (logged) interactions. This makes the inference of trust in real systems possible as interaction data are monitored and interpreted based on pre-specified rules. We have demonstrated the application of our trust model by supporting dynamic, trust-based partner discovery and selection mechanisms. This scenario is based on advanced interaction patterns in flexible compositions such as trusted delegations to achieve load-balancing and scalability in the Expert Web. Our future work will include the deployment and evaluation of the implemented framework in cross-organizational collaboration scenarios. This will be done within the EU FP7 COIN project focusing on collaboration in VOs. The emphasis of COIN is to study new concepts and develop tools for supporting the collaboration and interoperability of networked enterprises. The end-user evaluation in COIN will discover the usability of trusted expert discovery and balancing mechanisms.

Acknowledgement

This work is supported by the European Union through the FP7-216256 Project COIN.

References

[1] F. Skopik, D. Schall, S. Dustdar, Trustworthy interaction balancing in mixed service-oriented systems, in: ACM Symposium on Applied Computing (SAC), ACM, 2010, pp. 801–808.
[2] D. Schall, H.-L. Truong, S. Dustdar, Unifying human and software services in web-scale collaborations, IEEE Internet Computing 12 (3) (2008) 62–68.
[3] D. Artz, Y. Gil, A survey of trust in computer science and the semantic web, Web Semantics 5 (2) (2007) 58–71.
[4] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems 43 (2) (2007) 618–644.
[5] F. Skopik, D. Schall, S. Dustdar, The cycle of trust in mixed service-oriented systems, in: Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2009, pp. 72–79.
[6] M. Salehie, L. Tahvildari, Self-adaptive software: landscape and research challenges, ACM Transactions on Autonomous and Adaptive Systems 4 (2) (2009) 1–42.
[7] T.D. Huynh, N.R. Jennings, N.R. Shadbolt, An integrated trust and reputation model for open multi-agent systems, Autonomous Agents and Multiagent Systems (AAMAS) 13 (2) (2006) 119–154.
[8] T. Grandison, M. Sloman, A survey of trust in internet applications, IEEE Communications Surveys and Tutorials 3 (4) (2000).
[9] L. Mui, M. Mohtashemi, A. Halberstadt, A computational model of trust and reputation for e-businesses, in: Hawaii International Conferences on System Sciences (HICSS), 2002, p. 188.
[10] A. Agrawal, et al., WS-BPEL Extension for People (BPEL4People), Version 1.0, 2007 (specification available online).
[11] J. Golbeck, Trust and nuanced profile similarity in online social networks, ACM Transactions on the Web (TWEB) 3 (4) (2009) 1–33.
[12] F. Skopik, D. Schall, S. Dustdar, Start trusting strangers? bootstrapping and prediction of trust, in: International Conference on Web Information Systems Engineering (WISE), Springer, 2009, pp. 275–289.

[13] C. Dwyer, S.R. Hiltz, K. Passerini, Trust and privacy concern within social networking sites: a comparison of facebook and myspace, in: Americas Conference on Information Systems (AMCIS), 2007.
[14] M.J. Metzger, Privacy, trust, and disclosure: exploring barriers to electronic commerce, Journal of Computer Mediated Communication 9 (4) (2004).
[15] A. Abdul-Rahman, S. Hailes, Supporting trust in virtual communities, in: Hawaii International Conferences on System Sciences (HICSS), 2000.
[16] J. Sabater, C. Sierra, Social regret, a reputation model based on social relations, SIGecom Exchanges 3 (1) (2002) 44–56.
[17] S. Dustdar, T. Hoffmann, Interaction pattern detection in process oriented information systems, Data and Knowledge Engineering (DKE) 62 (1) (2007) 138–155.
[18] IBM, An architectural blueprint for autonomic computing, Whitepaper 2005.
[19] A. Jøsang, R. Ismail, The beta reputation system, in: Bled Electronic Commerce Conference, 2002.
[20] J. Patel, W.T.L. Teacy, N.R. Jennings, M. Luck, A probabilistic trust model for handling inaccurate reputation sources, in: International Conference on Trust Management (iTrust), vol. 3477, Springer, 2005, pp. 193–209.
[21] Y. Wang, M.P. Singh, Formal trust model for multiagent systems, in: International Joint Conferences on Artificial Intelligence (IJCAI), 2007, pp. 1551–1556.
[22] M.A. Orgun, C. Liu, Reasoning about dynamics of trust and agent beliefs, in: IEEE International Conference on Information Reuse and Integration (IRI), 2006, pp. 105–110.
[23] D. Schall, Human interactions in mixed systems—architecture, protocols, and algorithms, Ph.D. Thesis, Vienna University of Technology, 2009.
[24] W.M.P. van der Aalst, M. Song, Mining social networks: Uncovering interaction patterns in business processes, in: International Conference on Business Process Management (BPM), vol. 3080, 2004, pp. 244–260.
[25] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[26] N. Griffiths, A fuzzy approach to reasoning with trust, distrust and insufficient trust, in: CIA, vol. 4149, 2006, pp. 360–374.
[27] J. Sabater, C. Sierra, Reputation and social network analysis in multi-agent systems, in: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM, New York, NY, USA, 2002, pp. 475–482.
[28] W. Sherchan, S.W. Loke, S. Krishnaswamy, A fuzzy model for reasoning about reputation in web services, in: ACM Symposium on Applied Computing (SAC), 2006, pp. 1886–1892.
[29] H.-J. Zimmermann, Fuzzy set theory and its applications, third ed., Kluwer Academic Publishers, 1996.
[30] W.V. Leekwijck, E.E. Kerre, Defuzzification: criteria and classification, Fuzzy Sets and Systems 108 (2) (1999) 159–178.
[31] M. Srivatsa, L. Xiong, L. Liu, Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks, in: International World Wide Web Conference (WWW), ACM, 2005, pp. 422–431.
[32] Z. Gyngyi, H. Garcia-Molina, J. Pedersen, Combating web spam with trustrank, in: International Conference on Very Large Data Bases (VLDB), 2004, pp. 576–587.
[33] C.-N. Ziegler, J. Golbeck, Investigating interactions of trust and interest similarity, Decision Support Systems 43 (2) (2007) 460–475.
[34] Y. Matsuo, H. Yamamoto, Community gravity: Measuring bidirectional effects by trust and rating on online social networks, in: International World Wide Web Conference (WWW), 2009, pp. 751–760.
[35] M. Nowak, K. Sigmund, Evolution of indirect reciprocity by image scoring, Nature 393 (1) (1998) 573–577.
[36] F. Skopik, H.-L. Truong, S. Dustdar, VieTE—enabling trust emergence in service-oriented collaborative environments, in: International Conference on Web Information Systems and Technologies, INSTICC, 2009, pp. 471–478.
[37] D. Schall, C. Dorn, S. Dustdar, I. Dadduzio, Viecar—enabling self-adaptive collaboration services, in: Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2008, pp. 285–292.
[38] A. Reka, Barabási, statistical mechanics of complex networks, Reviews in Modern Physics 74 (2002) 47–97.
[39] B.L. Harrison, A. Cozzi, T.P. Moran, Roles and relationships for unified activity management, in: International Conference on Supporting Group Work (GROUP), 2005, pp. 236–245.
[40] P. Moody, D. Gruen, M.J. Muller, J.C. Tang, T.P. Moran, Business activity patterns: a new model for collaborative business applications, IBM Systems Journal 45 (4) (2006) 683–694.
[41] T.P. Moran, A. Cozzi, S.P. Farrell, Unified activity management: supporting people in e-business, Communications of the ACM 48 (12) (2005) 67–70.
[42] P.A. Balthazard, R.E. Potter, J. Warren, Expertise, extraversion and group interaction styles as performance indicators in virtual teams: how do perceptions of it's performance get formed? DATA BASE 35 (1) (2004) 41–64.
[43] N. Panteli, R. Davison, The role of subgroups in the communication patterns of global virtual teams, IEEE Transactions on Professional Communication 48 (2) (2005) 191–200.
[44] J. Breslin, A. Passant, S. Decker, Social web applications in enterprise, The Social Semantic Web 48 (2009) 251–267.
[45] S. Dustdar, Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams, Distributed and Parallel Databases 15 (1) (2004) 45–66.
[46] M. Amend, et al., Web Services Human Task (ws-humantask), Version 1.0, 2007 (specification available online).
[47] L.M. Camarinha-Matos, H. Afsarmanesh, Collaborative networks—value creation in a knowledge society, in: PROLAMAT, 2006, pp. 26–40.
[48] S.P. Marsh, Formalising trust as a computational concept, Ph.D. Thesis, University of Stirling, April 1994.
[49] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context aware systems, International Journal of Ad Hoc Ubiquitous Computing 2 (4) (2007) 263–277.
[50] S.W. Loke, Context-aware artifacts: two development approaches, IEEE Pervasive Computing 5 (2) (2006) 48–53.
[51] D. Kovac, D. Trcek, Qualitative trust modeling in soa, Journal of Systems Architecture 55 (4) (2009) 255–263.
[52] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, K. Nahrstedt, A trust management framework for service-oriented environments, in: International World Wide Web Conference (WWW), 2009.
[53] Z. Malik, A. Bouguettaya, Reputation bootstrapping for trust establishment among web services, IEEE Internet Computing 13 (1) (2009) 40–47.
[54] F. Kerschbaum, J. Haller, Y. Karabulut, P. Robinson, Pathtrust: A trust-based reputation service for virtual organization formation, in: International Conference on Trust Management (iTrust), 2006, pp. 193–205.
[55] Y. Zuo, B. Panda, Component based trust management in the context of a virtual organization, in: ACM Symposium on Applied Computing, 2005, pp. 1582–1588.
[56] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: International World Wide Web Conference (WWW), 2004, pp. 403–412.
[57] P. Massa, P. Avesani, Trust-aware collaborative filtering for recommender systems, in: CoopIS, DOA, ODBASE, 2004, pp. 492–508.
[58] G. Theodorakopoulos, J.S. Baras, On trust models and trust evaluation metrics for ad hoc networks, IEEE Journal on Selected Areas in Communications 24 (2) (2006) 318–328.
[59] C.-N. Ziegler, G. Lausen, Propagation models for trust and distrust in social networks, Information Systems Frontiers 7 (4–5) (2005) 337–358.
[60] E.M. Maximilien, M.P. Singh, Toward autonomic web services trust and selection, in: International Conference on Service Oriented Computing, 2004, pp. 212–221.