Adaptive Information Disclosure in a Dynamic Web of Trust

Under Review for Publication

Vienna University of Technology
Information Systems Institute
Distributed Systems Group

Florian Skopik, Daniel Schall,
Schahram Dustdar
lastname@infosys.tuwien.ac.at

TUV-1841-2010-03 April 19, 2010

Dynamic interactions in complex service-oriented systems increasingly demand for automated adaptation techniques to optimize resource discovery and selection. We describe a science collaboration scenario that applies mechanisms for adaptive information sharing, and introduce an adaptation approach that accounts for emerging trust relations based on varying interaction behavior of collaboration partners. This behavioral trust provides an intuitive grounding for customization and optimization of member compositions and sharing policies. As people prove their reliable and dependable behavior in jointly performed activities, they become increasingly considered as invaluable partners. We describe the foundational concepts, including support for ad-hoc and self-managed collaboration scenarios, and dynamic trust determination supported by SOA concepts. We highlight major concerns of trust management in highly dynamic networks and deal with temporal aspects such as supporting the emergence of trust, efficient update mechanisms, and aging of relations. Furthermore, we present a prototype implementation of a trust-based sharing framework, and evaluate its applicability from a system's perspective and end-user's view.

Keywords: information disclosure, collaborative environment, service-orientation, dynamic trust, behavior monitoring

Argentinierstr. 8/184-1
A-1040 Vienna, Austria
phone: +43 1 58801-18402
fax: +43 1 58801-18491
URL: http://www.infosys.tuwien.ac.at/

Adaptive Information Disclosure in a Dynamic Web of Trust

Florian Skopik*, Daniel Schall, Schahram Dustdar

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

Abstract

Dynamic interactions in complex service-oriented systems increasingly demand for automated adaptation techniques to optimize resource discovery and selection. We describe a science collaboration scenario that applies mechanisms for adaptive information sharing, and introduce an adaptation approach that accounts for emerging trust relations based on varying interaction behavior of collaboration partners. This behavioral trust provides an intuitive grounding for customization and optimization of member compositions and sharing policies. As people prove their reliable and dependable behavior in jointly performed activities, they become increasingly considered as invaluable partners. We describe the foundational concepts, including support for ad-hoc and self-managed collaboration scenarios, and dynamic trust determination supported by SOA concepts. We highlight major concerns of trust management in highly dynamic networks and deal with temporal aspects such as supporting the emergence of trust, efficient update mechanisms, and aging of relations. Furthermore, we present a prototype implementation of a trust-based sharing framework, and evaluate its applicability from a system's perspective and end-user's view.

Keywords:
information disclosure, collaborative environment, service-orientation, dynamic trust, behavior monitoring

1. Introduction

Over the past years, the Web has transformed from a pool of statically linked information to a people-centric Web. Information emerges bottom-up from a multitude of sources including social networks, online forums, or millions of news feeds [1]. Recently, the problem of careless personal information disclosure in today's interlinked online society has been clearly revealed. Especially, the combination of the willingness to 'friend' total strangers in Facebook[1], and open broadcasting about one's holiday plans, whereabouts, purchases, home interiors and other personal information in Twitter[2], are unrecognized threats. In some recent articles on the Web, e.g., see [2, 3], the authors discuss to what extent shared personal information of (naive) users may be exploited. However, not only in open, but also in closed virtual communities, information sharing is a delicate matter. Consider a traditional enterprise collaboration environment, where individuals need to share information to perform their tasks. In such scenarios, teams of people are formed that deal with single steps of project processes. The project management office assigns roles to each participating person, indicating not only their functional properties and responsibilities (e.g., manager, contributor, assistant, observer etc.), but also their (access) rights to centrally managed information and project artifacts. For instance, a sophisticated access rights management determines who may read, write, modify or create documents in the scope of a specific project. This is mostly controlled by roles assigned to people and their membership in certain user groups respectively. In contrast to this traditional and mainly static access rights management, we focus on novel approaches suitable for flexible ad-hoc collaboration systems. Especially in environments that are structured in flat organizational hierarchies, teams are mostly not created in advance, but rather establish themselves (*emerge*) based on successful collaborations [4]. In virtual communities, where people dynamically interact to perform activities, reliable and dependable behavior promotes the emergence of trust. As collabo-

*Corresponding author
Email addresses: skopik@infosys.tuwien.ac.at (Florian Skopik), schall@infosys.tuwien.ac.at (Daniel Schall), dustdar@infosys.tuwien.ac.at (Schahram Dustdar)

[1] http://www.facebook.com/

[2] http://twitter.com/

rations are increasingly performed online, supported by service-oriented technologies, such as communication-, coordination-, and resource management services, interactions have become observable. Some of the key challenges in mixed service-oriented systems comprising human actors and software services include:

- *Discovery of Network Members and Resources.* In many networks, for example social networks, the discovery and selection process relies on matching of user profiles and resource features that are mainly static. Utilizing periodically updated trust relations better accounts for varying user preferences and avoids lookup based on outdated information.

- *Access to and Sharing of Information.* Traditional approaches to access rights management are based on manually assigned static user roles. In dynamic environments, the user is often not able to keep track of configurations such as dynamically changing roles.

- *Coordination and Compositions.* Especially in flexible environments, compositions of humans and services cannot only rely on static structures, but have to be flexibly adapted based on their runtime behavior.

- *Interaction Policies and Patterns.* Policies and interaction patterns describe and restrict communication paths between network members. Therefore, periodic adaptation upon ongoing collaborations enable optimizations according to the outcome of interactions.

We demonstrated the inference of trust based on monitoring and analyzing fundamental interactions earlier [4, 5, 6]. In this paper, we study novel methodologies to flexible information sharing that account for the dynamics in self-organized ad-hoc collaboration environments. Our approach enables adaptive information sharing based on prior interaction behavior, and therefore, based on dynamically changing trust relations between actors. Similar to offline collaboration, we assume that people are encouraged to share more information with well-known and frequently interacting collaboration partners. As a typical use case, consider an academic research community. The single participants, such as university members and employees of national research labs, discuss novel research ideas, and exchange data via information and communication services. As researchers benefit from collaborations and work successfully together, e.g., resulting in higher paper output and accepted project proposals, trust in each other's competencies will emerge. Hence, people that have proven their cooperative, reliable, and dependable behavior in prior collaborations, are increasingly considered as invaluable partners. Our approach accounts for this behavior to decide the extent of shared personal information.

Trust models are applied in a wide variety of complex networks, such as service-oriented architectures and in social- and collaborative environments. The concept of trust facilitates the selection of network members and influences future interactions such as the exchange of information. Trust management systems that support such communities apply computational models to determine beneficial relations grounded in prior collaboration successes and the mutual fulfillment of user requirements. In contrast to fundamental quality-of-service (QoS) models, computational trust accounts for individual member behavior that is impacted by and closely linked to social influences, context awareness and personal preferences [4]. In social and collaborative environments, trust-based adaptation, such as (re-)selecting collaboration partners, are indispensable if members do not interact reliably or fail in performing certain activities.

Our contributions in this paper are as follows:

- *The Concept of Trusted Information Sharing.* We introduce a science collaboration scenario that motivates the need for trusted information sharing [7] and adaptive information disclosure in today's Web-based collaborations.

- *Dynamics in Social and Behavioral Trust Models.* We recapitulate system-managed trust models relying on interaction monitoring and behavior metrics interpretation [4, 5, 6]. We extend our model with mechanisms to effectively and efficiently cope with inherent trust dynamics due to changing actor behavior and user requirements in flexible collaboration environments.

- *Sharing Framework for Service-oriented Collaboration Networks.* We discuss an end-user perspective and show the prototype implementation of a sharing portal. Besides that, we highlight design decisions of our service-oriented sharing framework and outline its mode of operation.

- *Evaluation and Discussion*. We perform experiments to demonstrate the efficiency and effectiveness of our approach to address trust dynamics enabling flexible trust-based information sharing.

The remainder of the paper is organized as follows. Section 2 covers important related work including context-aware interaction models, trust models, and engineering of trust in service-oriented systems. Section 3 discusses a science collaboration scenario to draw the fundamental challenges and motivate our work. The following Section 4 deals with the aspects of supporting automatic trust emergence in general. In Section 5 we discuss novel concepts to cope with trust dynamics and its temporal properties regarding emergence, update and aging of relations. Then, we highlight the realization of a sharing portal in Section 6, discuss its design and implementation in Section 7, and evaluate its applicability in Section 8. Finally, Section 9 concludes the paper and gives an outlook of our future work.

2. Background and Related Work

We structure related work in three topics relevant for our adaptive, trust-based information disclosure approach. First, we discuss context and interaction models since context-aware, flexible interaction models are needed to cope with the increasing complexity of collaborations and distribution of people and services. Second, we believe that the key concept for sharing information is *dynamic trust*. In particular, people tend to share more information with others who have proven their reliable, dependable and secure collaboration behavior before. Third, since collaborations become increasingly complex and dynamic, manual adaptations of access rights are not sufficient any longer. Automatically managed trust relations are the ideal grounding to reflect these dynamics and an intuitive means to adapt access rights to resources automatically.

Context-aware Interaction Models. Context has been at the center of many research efforts. In computer science the definition given by Abowd et al. [8] is amongst the most adopted ones. Context-aware computing focusing on modeling and sensing of context can be found in [9, 10]. Traditional context models typically focus on the context of a person such as location, devices, presence information, time, and action. Some models include also user preferences specific to services such as cost, speed, QoS, and mobility [11].

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 [12]. They identify patterns of complex activities, which are then used to derive relationships and activity patterns [13, 14]. The potential impact of activity-centric collaboration is highlighted [15, 16] with special focus on the value to individuals, teams, and enterprises. Studies on distributed teams focus on human performance and interactions [17, 18], even in *Enterprise 2.0* environments [19]. Mixed service-oriented systems target flexible interactions and compositions of Human-Provided and software-based services [20]. This approach is aligned with the vision of the Web 2.0, where people can actively contribute services. In such networks, humans may participate and provide services in a uniform way by using the HPS framework [21]. A similar vision is shared by [22] who defines *emergent collectives* which are networks of interlinked valued nodes (services). In such collectives, there is an easy way to add nodes by distributed actors so that the network will scale.

Behavioral and Social Trust Models. Marsh [23] 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 [24, 25, 26], which outline common concepts of trust, clarify the terminology and describe the most popular models. From the many existing definitions of trust, those from [25, 27] describe that trust relies on previous interactions and collaboration encounters, which fits best to our highly flexible environment. Context dependent trust was investigated by [23, 24, 25, 26].

Depending on the environment, trust may rely on the outcome of previous interactions [6, 27], *and* the similarity of interests and skills [28, 29, 30, 31]. 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* [27] 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 has been studied before, e.g., in [32] and [33]. Trust propagation models [34, 35, 36, 37] are intuitive methods to predict relations where no personal trust emerged; e.g., transitive recommendations.

The interplay of trust and privacy has been studied in the areas of social networks [38] and e-commerce [39]. Especially the latter work concludes that trust is strongly related to information disclosure, and thus, privacy. Articles on the Web [2, 3] discuss the exploitation of personal information and underline the role of privacy. Marsh discussed in [40] how trust models enhance information sharing among community members.

Engineering Self-adaptation in SOA. Enhanced flexibility of complex systems is introduced by establishing a cycle that feeds back environmental conditions to allow the system to adapt its behavior. This MAPE cycle [41] is considered as one of the core mechanism to achieve adaptability through self-* properties. While autonomic computing allows for autonomous elements and applies these principles to distributed systems, current research efforts left the human element outside the loop. Based on the observed context of the environment, different adaptation strategies can be applied [42] to guide interactions between actors, the parameters of those strategies, and actions to prevent inefficient use of resources and disruptions. In the context of multi agent systems (MAS), self-configuring social techniques were introduced in [43]. A major challenge in adaptation and self-configuration is to dynamically find the most relevant adaptation parameter. Research relevant to this issue can be found in [44].

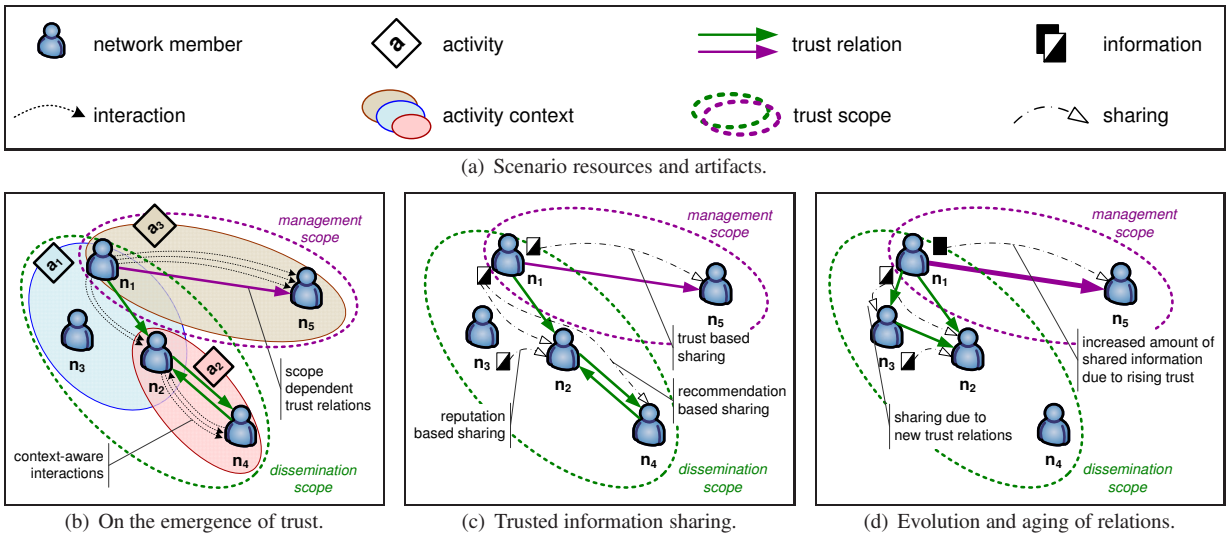
Recently, trust in service-oriented systems has become a very important research area. SOA-based infrastructures are typically distributed comprising a large number of available services and huge amounts of interaction logs. Therefore, trust in SOA has to be managed in an automatic manner. A trust management framework for service-oriented environments has been presented in [45, 46, 47], 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. The technical realization of *trusted information sharing* in the introduced collaboration network is related to *selective dissemination of information* (SDI) [48, 49]. SDI deals with questions regarding which (parts of) data are shared with others, and mechanisms to disseminate data. We adopted concepts of SDI, such as the representation of information through XML, or mechanisms to process XML-based data.

3. The Science Collaboration Scenario

A typical environment for applying trusted information sharing is a *science collaboration network*. It comprises scientists, members from national and international research labs, and experts from the industry. Collaboration is supported by modern service-oriented architectures that realize centralized people registries and profile management, communication services, and data sharing facilities. Network members collaborate to address challenging research questions and to reach higher impact of scientific disseminations. They conduct joint project proposals, perform distributed software prototyping, and data analysis and visualization. Furthermore, certain participants can provide their support in a service-oriented manner. For instance, they offer document review services, or data analysis services, and interact through precisely predefined interfaces. We utilize the previously introduced Human-Provided Services (HPS) framework [20] to embed humans acting as services using SOA concepts. This includes WSDL descriptions of interfaces, central registries, SOAP-based interactions, and sophisticated logging facilities.

3.1. Emerging Trust Networks

We demonstrated the (semi-)automatic flexible determination of trust in the above-mentioned service-oriented collaboration environment in detail earlier [4, 6]. Briefly, our approach relies on the observation of fundamental interactions, such as SOAP-based communication, coordination or execution messages. People interact and use services when conducting activities. Figure 1(b) depicts this fundamental concept. Network members collaboratively perform activities of different types. These activities structure relevant contextual information, including involved actors, goals, temporal constraints, and assigned resources. So, we conclude that an activity holistically captures the context of interactions between participants [16]. Several activity contexts are aggregated to uniform scopes, e.g., all activities of a specific type (activity scope), or all activities belonging to a certain project (project scope). Trust emerges from interactions and manual ratings of collaboration partners within those scopes. For instance, trust can rely on the responsiveness and reliability of collaboration partners, as well as on their collected experiences and skills. As shown in Figure 1(b), trust is represented by a directed relation from one network member n_i (the *trustor*) to another one n_j (the *trustee*), and relies on prior cooperative behavior in a given scope. These trust relations are determined by periodically analyzing and interpreting observed interactions and ratings of partners. For example, the collaboration of network members n_1, n_2, n_3, and n_4 in different scientific dissemination activities a_1 and a_2, leads to the establishment of trust in one uniform 'dissemination scope'. Finally, a scale-free complex network emerges from coopera-



(a) Scenario resources and artifacts.

(b) On the emergence of trust.

(c) Trusted information sharing.

(d) Evolution and aging of relations.

Figure 1: Information sharing upon dynamically changing trust relations in service-oriented collaborations.

tions in typical research collaborations as investigated by [50].

3.2. On Trusted Information Sharing

In a science collaboration network scenario, understandably no member will share novel, yet unpublished, ideas carelessly. However, information sharing is essential to discover new collaboration opportunities. The challenge is to enable sensitive information sharing, that adapts and restricts the view on information with respect to changing trust relations. Therefore, we introduce the concept of *trusted information sharing*. This concept provides the means to share information, e.g., paper drafts, recently submitted papers, or project documentation, only with trusted network members who have demonstrated their reliable and dependable behavior before. In this case, trust reflects a probability measure of future collaboration successes, and therefore, potential benefits from collaborations.

As depicted in Figure 1(c), trusted information sharing is bound to trust scopes. For instance, if member n_1 established trust in n_5 in the management scope (because they jointly performed several project management activities successfully), n_5 is allowed to access n_1's data about referees' contact details, planned future projects, and personal organizational details. However, no information dedicated to other scopes, such as scientific dissemination, is shared. Hence, information sharing is restricted to mandatory information in particular scopes.

As trust relations emerge dynamically based on interaction behavior of people, the amount of shared information is periodically adapted by the system and, in the optimal case, needs no further manual intervention of users. However, this approach works best in environments with flat (or practically no) hierarchies, where people may decide completely on their own about conditions for information sharing. In enterprise collaborations, with pre-determined communication paths and static role models, mechanisms that override trust-based sharing are required. But here, we focus on the depicted science collaboration network that consists of people with equal roles, rights and aims. We identified three fundamental trust concepts to enable trusted information sharing in the described environment:

Sharing based on Personal Trust Relations. Activity relevant artifacts are shared in a scope to different extent (views), according to the degree of trust between network members. For instance, in Figure 1(c) n_1 grants access to n_5 to information in the management scope.

Sharing based on Recommendations. In case of sparse trust networks, or low connectivity of certain members, sharing based on personal relations only is limited. Second-hand opinions, called recommendations, are utilized to overcome this problem. For instance, n_1 trusts n_2, and n_2 trusts n_4 because of successful previous collaborations in the dissemination scope. If these successes rely on the compatibility of each member's working style, there is a high probability that n_1 might establish trust to n_4 upon future interactions (for transitive trust propagation see [34]). Hence, to facilitate the establishment of trust relations, n_1 is encouraged to share pieces of information with the unknown

member n_4. Sharing of data, such as parts from the personal profile, introduces n_1 to n_4 and supports the bootstrapping of future collaborations [28].

Sharing based on Reputation. If network members are trusted by several partners in the same scope, (i.e., they have more than one trustor), reputation can be determined. For instance, n_2 is trusted by n_1 and n_4. Therefore, network member n_3, who has not established trust in others yet, can rely on this reputation (inferred from single trust relations). So, n_3 can either allow n_2 to access parts of his personally managed information (passive sharing), or by pushing information towards n_2 (active sharing).

3.3. Evolution and Aging of Trust

Since network members may change their interaction behavior over time, for instance, their goals and priorities shift or they start to develop interests in new fields, trust relations have to be altered too. However, trust relations can become even closer through successful long-term collaborations. These dynamics are indicated from Figure 1(c) to Figure 1(d). Here, trust from n_1 to n_2 has been increased, thus n_1 grants n_2 even more access to his personal files (see the filled document symbol). While trust from n_1 in n_3 and n_3 in n_2 has emerged due to closer collaboration, the relations from and to n_4 have been removed.

Especially in our science collaboration scenario, it is inevitable to consider these trust dynamics. Imagine, someone suddenly begins to behave unreliable, e.g., does not answer support requests or does not fulfill his assigned activities any longer. In that case also the access to critical information has to be restricted. In this paper we discuss approaches to detect misleading behavior changes to guarantee timely updates of relations in a managed *Web of Trust*. However, not only existing relations are adapted, but new relations are introduced and outdated relations removed.

4. On the Emergence of Trust Relations

In contrast to a common security perspective, we define (social) trust to rely on the interpretation of previous collaboration behavior [6] and may additionally consider the similarity of dynamically adapting interests [29, 30]. Especially in collaborative environments, where users are exposed to higher risks than in common social network scenarios [38], and where business is at stake, considering social trust is essential to effectively guide interactions [39]. Hence, we define trust as follows [6, 25, 27]:

> *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.*

Previously, we introduced a conceptual approach for determining trust based on interactions that we call the *Cycle of Trust* [4]. The cycle, adopting the MAPE concept [41], 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. Because this approach (Figure 2) is fundamental for the design of our trust model, we shortly recapitulate its mode of operation.

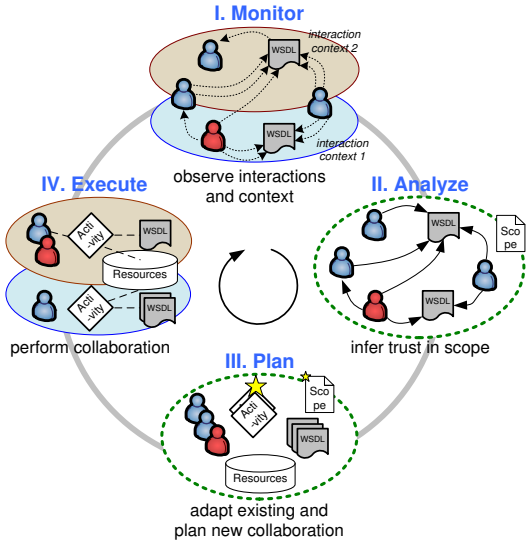


Figure 2: The Cycle of Trust.

In the *Monitoring Phase* the trust management system observes interactions between humans and services, including their types, context and success. In the *Analyzing Phase* the interactions are used to infer trust relationships. For this purpose, interaction metrics are calculated and interpreted using scope-aware 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, therefore, the loop is closed.

4.1. Interactions in Service-oriented Collaborations

Interactions in flexible collaboration environments on the Web are typically not modeled in advance, but interaction partners are discovered dynamically based on collected experiences from previous interactions. Here, we discuss the technical grounding to capture interactions and their context.

4.1.1. Interaction Context Model

Community members interact to reach a predefined goal. For instance, they request support, exchange information, delegate tasks, and coordinate actions to perform certain activities. Therefore, interactions always take place within certain contexts. Figure 3 shows the applied context model, where `actors`, described by their `profiles`, perform `activities`. Activities reside in more abstract `scopes`, e.g., all activities of a specific type (activity scope), or all activities belonging to a certain project (project scope). For instance, supporting the creation of white box test cases resides in a software development scope. Furthermore, actors are linked to collaboration partners in the network. These `relations` that are bound to scopes are characterized by various `metrics` that rely on previous interactions.

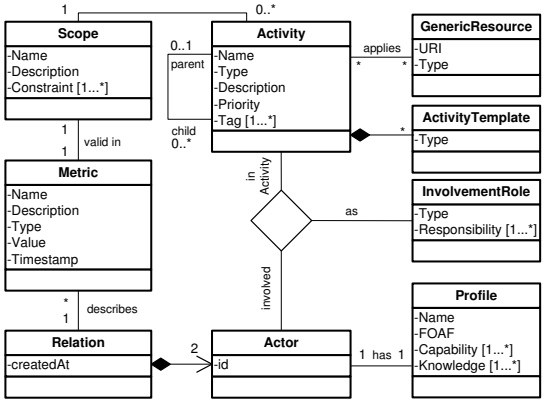


Figure 3: Interaction context model: Linked Actors perform activities that reside in scopes.

4.1.2. Interactions in SOA

Not only service interactions, but also human interactions may rely on SOAP (e.g., see Human-Provided Services [20] and BPEL4People [51]), which is the state-of-the-art technology in service-oriented environments, and well supported by a wide variety of software frameworks. This fact enables the adoption of various available monitoring and logging tools for mixed service-oriented systems. The XML-based structure of SOAP messages is well-suited for message header extensions, such as addressing and routing information, and annotation with contextual elements (e.g., activity identifier). These mechanisms allow for context-aware interaction metric calculation, for instance, reliability, responsiveness, collected experience, and costs in a predefined scope. We apply an interpretative inference of trust based on these metrics. This approach is context dependent, so in different domains and use cases the impact of the same metrics varies. As interaction behavior changes over time, trust alters too. Therefore, trust deems to be an intuitive grounding for flexible adaptation techniques in mixed service-oriented systems.

```
<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="2010-03-21"/>
  <vietypes:msgflags priority="urgent"/>
  <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>ACM taxonomy for my paper?</rfs:subject>
   <rfs:requ>What ACM categories fit to my paper?</rfs:requ>
   <rfs:resource>
      <vietypes:resource type="paperdraft" uri="http://..."/>
   </rfs:resource>
   <rfs:keywords>research, paper, ACM taxonomy</rfs:keywords>
  </hps:Request>
 </soap:Body>
</soap:Envelope>
```

Listing 1: Simplified RFS via SOAP example.

Listing 1 depicts an example interaction with various header elements. The most important extensions are (see [6] for details on the implementation):

- `Timestamp` capture the actual creation of the message and is used to calculate temporal interaction metrics, such as average response times.
- `Message flags`, including priority of messages.
- `Activity uri` describes the context of interactions (see [16] for activity model).
- `MessageID` enables message correlation, i.e., to properly match requests and responses.
- `WS-Addressing` tags, besides MessageID, are used to route requests through the network.

The SOAP body transports the actually exchanged message. In this example a request for support (RFS) [52] shows how one actor requests some help from another one in the science collaboration scenario.

4.1.3. Social Relation Metrics and Network Model

Relation metrics describe the links between actors by accounting for (i) recent interaction behavior, (ii) profile similarities (e.g., interest or skill similarities), (iii) social and/or hierarchical structures (e.g., role models). However, we argue that social trust relations largely depend on personal interactions.

We model a community of actors with their social relations as a directed graph $G = (N, E)$, where a node $n_i \in N$ denotes a network member, and an edge $e_i j \in E$ reflects a relation from n_i to n_j. Since interaction behavior (that determines relations) is usually not symmetric, actor relations are represented by *directed links* in our network model. For instance, an actor n_i might serve n_j reliably, but not the other way around.

Metrics, calculated on top of observed SOAP interactions and modeled profile data, annotate the relations between network members in G. The following classes of metrics are managed (i) *Interaction Metrics* describe the interaction behavior as explained before, such as an actor's responsiveness and reliability in distinct scopes. (ii) *Similarity Metrics* provide information about skill-, feature-, or expertise similarities, depending on the type of actors. Similarity metrics are an indicator for replaceability of actors. (iii) *Trust Metrics* are interpreted from interaction- and similarity metrics, e.g., personal trust, symmetry of trust relations (bidirectional trust) and trust trends in certain time intervals. See next subsection for details on their determination. (iv) *Collaboration Metrics* 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 [27]. Furthermore, edge metrics can be aggregated to calculate collaboration metrics; for instance, an average value of someone's responsiveness or availability. (v) *Group Metrics* provide information about average values and distribution of node- and edge metrics in a community, therefore, they are a valuable means to determine a metric value relative to others in the same group.

Table 1 shows some example metrics that may describe actor relationships, and that are calculated from logged SOAP calls and user-provided profile information. Note, as described before, these metrics are determined for particular scopes. The availability of an actor can be high in one scope, but much lower in another one.

Table 1: Example metrics describing actor relations.

metric name	range	description
availability	[0,1]	ratio replied to all incoming requ.
response time	[0,96]	average response time in hours
success rate	[0,1]	amount of successful activities
reciprocity	[-1,1]	ratio of served to sent requests
experience	[0,∞]	number of served support requ.
manual reward	[0,5]	manual feedback scores
costs	[0,5]	price for requesting support
interest similarity	[0,1]	interest profile overlap

In this paper, we make use of the following metrics to infer trust in the science collaboration scenario:

Interest Similarity *isim***.** This metric determines the overlap of actor interests, which is an important measure to find collaboration partners in research. We manage keywords used by paper authors n_i and n_j as interest profile vectors $\mathbf{p}_{n_i}$ and $\mathbf{p}_{n_j}$ respectively (see [29] for details), and determine the similarity of profiles through the cosine between their profile vectors (Eq. 1). The result is a value between 0 (no overlap) and 1 (full overlap).

$$isim(n_i, n_j) = cos(\mathbf{p}_{n_i}, \mathbf{p}_{n_j}) = \frac{\mathbf{p}_{n_i} \cdot \mathbf{p}_{n_j}}{|\mathbf{p}_{n_i}||\mathbf{p}_{n_j}|} \quad (1)$$

Reciprocity *recpr***.** A typical social behavior metric is reciprocity [27] that reflects the ratio between obtained and provided support in a community. Let $REQ(n_i, n_j)$ be the set of n_i's sent support requests to n_j, and $RES(n_i, n_j)$ the set of n_i's provided responses to n_j's requests. Then we define reciprocity in $[-1, 1]$ as in Eq. 2; hence, 0 reflects a balanced relation of mutual give and take.

$$recpr(n_i, n_j) = \frac{|REQ(n_i, n_j)| - |RES(n_i, n_j)|}{|REQ(n_i, n_j)| + |RES(n_i, n_j)|} \quad (2)$$

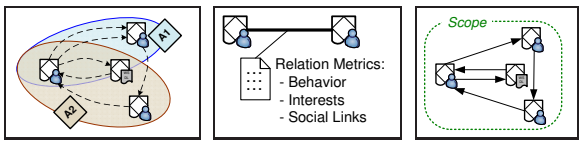
Availability *avail***.** This metric describes n_i's availability for n_j's requests, i.e. the amount of answered requests. The result of Eq. 3 is a value in [0, 1].

$$avail(n_i, n_j) = 1 - \frac{|REQ(n_j, n_i)| - |RES(n_i, n_j)|}{|REQ(n_j, n_i)|} \quad (3)$$

4.2. Trust Models and Management

The fundamental approach to automatic interaction-based trust inference is depicted in Figure 4. As motivated in the introduced use case, people interact to perform their tasks. This work is modeled as activities, that describe the type and goal of work, temporal constraints, and used resources. As interactions take place

in context of activities (Figure 4(a)), they can be categorized and weighted. Interaction logs are used to infer metrics that describe the relation of single actors (Figure 4(b)), such as their behavior in terms of availability and reciprocity.



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

Figure 4: Trust emerging from interactions: (a) interaction patterns shape the behavior of actors in context of activities; (b) (semi-) automatic rewarding of behavior and calculation of interaction metrics; (c) trust inference in scopes by interpretation of metrics.

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 or social relations (e.g., FOAF[3], XFN[4]), as well as service features and capabilities may influence the trust inference process. Finally, available relation metrics are weighted, interpreted, and composed by a rule engine. The result describes trust between the actors with respect to scopes (Figure 4(c)). For instance, trust relations in a scope 'scientific dissemination' could be interpreted from interaction behavior of actors in a set of paper writing activities.

4.2.1. Flexible Behavioral Trust and Reputation Model

The interaction behavior of a network member n_i toward n_j is described by various metrics $M(n_i, n_j)$. In this work, we use two different approaches to infer trust upon these metrics:

- *Arithmetic Calculation* is used to calculate trust by weighting metrics to obtain average values. This simple model is quite effective under certain assumptions such as simple interaction types and patterns.
- *Rule-based Interpretation* allows for more complex business rules to interpret metrics. For instance, actors need to reach certain scores of predefined metrics to be considered trustworthy.

Arithmetic Calculation. Interaction metrics are normalized to the interval $[0, 1]$ either according to predefined upper and lower bounds, or dynamically adapted according to the highest and lowest values in the whole community. Furthermore, weights need to be specified, either by users or system administrators. The weighted sum of selected metrics build the actual trust value $\tau^s(n_i, n_j) \in [0, 1]$, and reflects recent evidence that an actor behaves dependably, securely and reliably. In Eq. 4 trust is calculated from metrics $m_k \in M(n_i, n_j)$ that are weighted by w_k with $\sum_k w_k = 1$.

$$\tau^s(n_i, n_j) = \sum_{\forall m_k} m_k(n_i, n_j) \cdot w_k \tag{4}$$

Rule-based Interpretation. Interaction metrics are processed by individually configured fuzzy (E)CA rules ((event)-condition-action). These rules define conditions to be met by metrics M for interpreting trustworthy behavior, e.g., 'the reciprocity of the trustee must be *positive*' or 'a trustworthy software programmer must have collected at least *average* experiences in software integration activities'. The definition of the linguistic variables, i.e, the meaning of low, medium, high, positive, negative, balanced etc. is either (i) defined statically, or (ii) dynamically adapted by accounting for average values of members in the whole community. Rules reflect a user's trust perception, e.g., pessimists may demand for stricter trustworthy behavior, than optimists. So, these rules are either specified by the users, or globally by administrators. Note, only the individual definition of rules allows the expression of personalized *trust requirements*. However, individual rules highly increase the computational complexity for trust inference.

```
if isim is high and recpr is positive then τ is high
if isim is high and recpr is balanced then τ is med
if isim is high and recpr is negative then τ is med
if isim is med  and recpr is positive then τ is high
if isim is med  and recpr is balanced then τ is med
if isim is med  and recpr is negative then τ is low
if isim is low  and recpr is positive then τ is med
if isim is low  and recpr is balanced then τ is low
if isim is low  and recpr is negative then τ is low
```

Listing 2: An example rule base for given linguistic variables *interest similarity isim*, *reciprocity recpr*, and *trust* τ.

Listing 2 shows an example for such rule definitions used by a fuzzy rule engine. This rule set determines the aggregation of two metrics: *interest similarity* (*isim*) and support *reciprocity* (*recpr*). These two metrics are typical for a science collaboration as discussed in the motivating scenario of this paper. For instance, someone looking for beneficial collaborations will neglect costs or responsiveness of actors and focus more on their similarity of interests. After determining the linguistic representation of trust (i.e., low, medium, high, full) a real value in the interval $[0, 1]$ is obtained through defuzzification. However, the detailed description of the

[3]Friend-Of-A-Friend `http://xmlns.com/foaf/spec/`
[4]XHTML Friends Network `http://www.gmpg.org/xfn/`

utilized inference mechanisms have been studied in [6] and are out of scope of this paper that focuses on the temporal evolvement of existing relations.

4.2.2. Trust Emergence through Delegations

Actors in the science collaboration scenario may request support from potential collaboration partners. However, the request receiver does not need to process all requests directly. As in real life, they can delegate them to other members due to various reasons. For instance, an actor may be overloaded and therefore, not be able to process a request in time. On the other side an expert can be 'overqualified', therefore delegate requests to lower experienced but sufficiently skilled actors. Moreover, some actors may shield others from requests, e.g., only a team leader receives requests directly that s/he delegates then to team members.

We introduce a triad interaction pattern (Figure 5) that realizes delegations of requests within the same contextual or organizational scope, e.g., between actors in the same knowledge domain (of course, one actor may be 'located' in several scopes). The triad pattern is well known as *triadic closure* in social networks [53]. A *triad proxy* n_2 receives a request and forwards it to one of its well-trusted partners. In case of complex problems, a request can be split into sub-requests to reduce response times. Furthermore, the complete request can be delegated to more than one actor to increase reliability, i.e., the chance to get a suitable response. Final responses are not handled by the triad proxy. As all participating entities in this pattern belong to the same scope, e.g., knowledge domain, the finally serving actor(s) may respond directly to the requester. A typical use case is load balancing in teams of people with same roles.

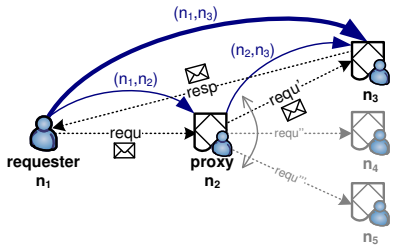


Figure 5: Interactions following the Triad Delegation Pattern.

From the requester n_1's point of view, the triad proxy n_2 receives reduced rewards for delegating but not processing the request. The actually supporting actor n_3 receive rewards from the triad proxy, because of accepting the delegated request. This reward is also reduced, because the originator of the request is not the triad proxy, and the triad proxy has limited interest in successfully processing the request (compared to one of his own requests). However, the requester honors the help provided by the actually supporting actor(s) through delegations equally compared to directly supporting actors.

The triad delegation pattern is an intuitive means to facilitate the emergence of new trust relations. For instance, because of n_2's delegation, n_3 gets introduced to n_1. If n_3 reliably serves n_1's request trust is built, and n_1 will consider n_3 as trustworthy collaboration partner when sending future requests.

5. Adaptive Trust Management

We argue that trust from a social perspective is neither identified in advance nor defined statically. It rather emerges dynamically upon interaction behavior of humans and services, and evolves over time. Sophisticated trust models need to account for these properties to reflect real situations as close as possible. Moreover, in highly flexible environments, interaction behavior may alter quickly and, therefore, the underlying trust model has to be updated in sufficiently short cycles. However, in large-scale networks with potentially thousands of participants, updating relations in short intervals is not feasible due to limitations of computational power. Hence, trust relations have to be updated and altered selectively.

Aging models for the WWW [54] describe common characteristic change rates of Web pages. Based on that knowledge, methods have been introduced that let search engines decide about suitable update intervals of their search indexes. If update intervals are too long, then outdated information is managed by search engines. However, for update intervals being too short, suitable usage of limited computational power and bandwidth is no longer guaranteed, and index entries are refreshed for pages that have not changed in the meanwhile.

Models that account for the dynamics of trust, are applicable in various kinds of flexible interaction environments, including the following:

- *SOA-based Service Networks.* Selecting the most beneficial services for compositions, by accounting not only for functional requirements, but also for reliability and dependability upon recent behavior, is a key research challenge in service-oriented computing. Especially, if facing large sets of potential services to be used, trust-supported discovery can considerably enhance the selection process.

- *Social Networks.* In today's social networks people declare their trusted friends manually in static lists, however, based on their interaction behavior, interest similarities and joint group memberships, trust can be predicted to some extent automatically [28, 31]. This feature relieves members from keeping track with the real situation and updating their friend lists manually. Also, recommendations of people, groups or items in social networks can be improved.

In this paper, we mainly address the following two issues of dynamic trust models resulting from interaction flexibility:

- *Efficiency of Trust Update Models* in terms of performance is realized by carefully selecting the most critical trust relations in a network to be refreshed in adaptive update cycles. Scheduling of updates fundamentally depends on the actors' interaction behavior, and the community's utility of frequent updates.

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

5.1. Challenges

Trust relations are not statically defined, but emerge and evolve over time. Thus, an efficient trust model must frequently refresh its data to keep track of the real situation. We model the following fundamental lifecycle phases of trust relations to account for their dynamic nature and temporal aspects: (i) *Trust Emergence* deals with introducing new trust relations upon ongoing interactions. (ii) *Trust Update* deals with refreshing existing relations based upon experiences made in recent interactions. (iii) *Trust Aging* deals with degrading and deleting outdated trust relations.

5.1.1. Trust Emergence

Several concepts of trust prediction exist to introduce new relations between actors that may develop trust in the future. Recent research shows, that there is a strong dependency between interest similarities and trust [28, 29, 30]. Furthermore, someone could recommend trusted collaboration partners to third persons, because of their distinguished expertise and reliable working style. However, there is no evidence that someone's partners will behave trustworthy toward third persons. Finally, this means that trust can only be reliably inferred from personal experiences, i.e., by analyzing interactions. We consider the following two fundamental types of trust emergence.

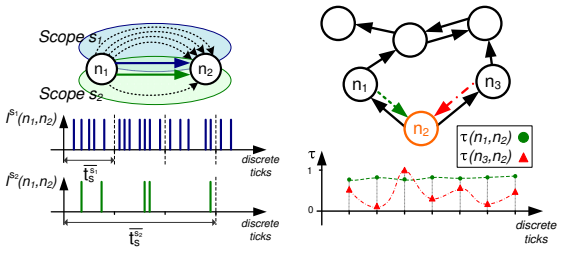
Trust Prediction. We predict trust based on recommendation and reputation mechanisms, and let especially newcomers enjoy initial trust. These trust relations given in advance are subject to updating and aging, and hence, are either weakened or strengthened over time.

Trust Inference. Trust is only built upon personal experiences from recent interactions and, therefore, only if there is a sufficient amount of interactions to reliably infer trust. We enable the application of this concept through delegations, where untrusted actors start interacting.

The choice for one or another method strongly depends on the environment. In the case of trust prediction, new relations emerge rather quickly, however, outdated relations require correction, i.e., removal after unsuccessful collaborations. Consider, applying trust formations upon prior evidence may be impossible since humans tend to interact predominantly with already well-known and trusted partners and newcomers are left out of collaborations.

5.1.2. Trust Update

Since the behavior of actors in a network may change due to various reasons, e.g., shift of interests, work overload, and search for new work opportunities, trust relations will alter as well. Hence, frequent synchronization with the real world is critical to computational trust models. For that purpose, the behavior of actors is sampled (i.e., observed through monitoring) in subsequent intervals and results are used to update trust relations. A major challenge is to determine the appropriate sampling intervals (e.g., see also [55]). Figure 6 visualizes two fundamental challenges of trust update mechanisms:



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

Figure 6: Challenges for trust update mechanisms.

Interaction Sparsity. In different scopes s_1, s_2 occur varying types and amounts of interactions. Since a larger amount of interactions is needed to detect trends in an actor's behavior (e.g., his responsiveness, availability), it is a challenging task to set the right size of sampling intervals ($\overline{t_s}$). Intervals that are too short prohibit reliably behavior analysis; however, if intervals are too long, sudden changes of behavior cannot be detected accordingly. The length of $\overline{t_s}$ mainly depends on the scope and the regular interaction behavior of actors therein.

Actor Uniformity. The uniformity describes the consistency of an actor (i) towards the same partner over time; (ii) towards different interaction partners. In Figure 6(b) n_2 behaves consistently trustworthy towards n_1, therefore, trust $\tau(n_1, n_2)$ remains high over several sampling intervals. However, n_2 changes dynamically his behavior toward n_3, and hence, n_3's trust in n_2 changes rapidly over time. Intuitively, in the second case of dynamically changing behavior, smaller sampling intervals are required to capture n_2's behavior changes, while in the first case, the sampling interval to refresh already well-known constant behavior can be longer. Apart the actors' interaction behavior, external adaptation requirements may influence the determination of appropriate sampling intervals; e.g., in the case of pre-defined upper time limits to react on changes in time.

5.1.3. Trust Aging

If the amount of interactions between two actors falls below a certain threshold, or two actors completely stop interacting, relations undergo an aging process. Since in this phase no further evidence occurs for reliably interaction behavior, relations are not updated any longer. Therefore, trust relations will degrade to a neutral state and are finally removed from the graph G.

Intuitively, well established and consolidated long-term relations mature slower compared to fragile short-term relations. Hence, strengthened long-term relationships are able to bridge longer 'interaction gaps', while short-term relations disappear faster.

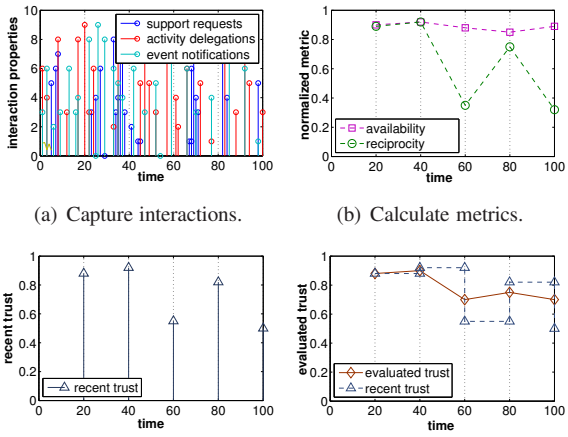
5.2. Adaptive Trust Update and Aging Models

In our model, the selection of relations to be updated and update intervals relies on two influential factors (i) the variance of user behavior, reflected by the dynamics of interaction metrics, (ii) the sparsity of interaction, i.e., a certain amount of interactions is required to reliably calculate interaction metrics.

Note, for the initial establishment of trust relations interactions are not mandatory, but relations can be introduced manually. This is a sufficient assumptions, as in real environments actors are selected based on recommendations or reputations and, therefore, enjoy initial trust. However, once established, manually introduced trust relations are automatically updated by the system considering trust aging parameters. The advantage of manually introduced relations is the reduced effort when processing interaction logs. In this case, only interactions between actors that also share a trust relation need to be handled.

5.2.1. Fundamental Update Mechanisms

Figure 7 summarizes the fundamental mode of operation of our temporal trust management approach. Interactions from n_i to n_j (a), occurring between two sampling instants (in this example every 20 ticks), are utilized to calculate interaction metrics $M(n_i, n_j)$ (b). These metrics describe actor n_j's behavior toward n_i in scope s, and is inferred in consecutive sampling intervals $\overline{t_s}$; for instance, n_j's availability to n_i's requests as well as its reciprocity. In the given example, the availability remains high, while n_j's reciprocity toward n_i is unsteady. We assume trust $\tau^s(n_i, n_j)$ relies on both metrics. Therefore, in (c), recent trust $\widehat{\tau^s}(n_i, n_j)$, grounded in previous interaction behavior of n_j toward n_i in time interval $\overline{t_s}$, is inferred. This $\widehat{\tau^s}(n_i, n_j)$ is visualized in Figure 7(c) at the sampling instants.



(a) Capture interactions. (b) Calculate metrics.

(c) Interpret trust. (d) Support trust evolution.

Figure 7: Illustrative example for updating trust based on recent interaction behavior.

Evolving long-term trust τ_i^s (see Figure 7(d)) is updated periodically in successive time intervals t_i (e.g., days in our motivating scenario), 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^s_{i-1}, with newer ones, but merge them accordingly. For this purpose we apply the concept of exponential moving average (EMA)[5], to smoothen the sequence of calculated trust values as shown in Eq. 5. Using this method, we are able to adjust the importance of the most recent behavior (leading to $\widehat{\tau}^s$) compared to historical values. The smoothing factor $\alpha \in [0, 1]$, can be dynamically selected. The impact of the most recent trust values $\widehat{\tau}$ on well established long-term relations might be lower than on recently emerged and still fragile short-term relations. For the reason that long-term relations are normally based on large sets of previous experiences and sporadic short-term behavior changes, e.g., sporadic unreliability, may not have major impact. Indeed, this behavior is subjective, and our model can not dictate the application of this feature, but provides the means to cover such situations appropriately.

$$\tau^s_i = \alpha \cdot \widehat{\tau}^s + (1 - \alpha) \cdot \tau^s_{i-1} \tag{5}$$

5.2.2. Adaptive Sampling

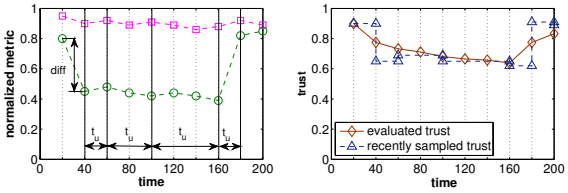
The fundamental approach simply updates τ_i at each time tick t_i, and the interval between instant t_i and t_{i+1} is always $\overline{t_s}$. However, in most real situations an adaptive sampling interval $\overline{t_s}$ is desired due to two reasons: (i) interaction sparsity, and (ii) actor (non-) uniformity (see Section 5). Intuitively, trust relations in erratic actors that change their behavior quickly and dynamically have to be updated more often, than the relations to stable actors. From a performance perspective, longer update cycles of stable connections allows the system to focus on unstable connections. Hence, while in the fundamental case we set the update interval in a particular scope s to $\overline{t^s_u} = \overline{t_s}$ (equal to the system sample interval), we introduce now an approach to adapt $\overline{t^s_u}$ dynamically within the limits according to Eq. 6.

$$\overline{t^s_{u_{min}}} \leq \overline{t^s_u} \leq \overline{t^s_{u_{max}}} \tag{6}$$

Both limits are pre-configured and determined by the interaction sparsity in the scope of trust. Furthermore, $\overline{t^s_{u_{min}}} = \lambda_1 \cdot \overline{t_s}$ and $\overline{t^s_{u_{max}}} = \lambda_2 \cdot \overline{t_s}$ and $\lambda_1 \leq \lambda_2$ for $\lambda_1, \lambda_2 \in \mathbb{N}$. The basic challenge is to find appropriate update intervals $\overline{t^s_u}$, in terms of efficiency and effectiveness of trust management. Remember, although static relations do not need frequent updates, sudden behavior changes must not be neglected. The mode of operation of our adaptive approach is depicted in Figure 8.

We interpret actor behavior, reflected by metrics M as a continuous 'signal' that is sampled from interactions

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



(a) Triggering metric changes. (b) Dynamic trust evaluation.

Figure 8: Illustrative example of adaptive trust update intervals through triggering sudden behavior (metric) changes.

in consecutive time intervals $\overline{t_s}$. Therefore, metrics reflect the changeability of an actor's behavior. Figure 8(a) shows the temporal evaluation of two interaction metrics, e.g., availability and reciprocity. While the values of one metric are nearly constant over time, the other suddenly drops at time tick 40, remains low, and increases again at tick 180. We detect such rapid changes with precisely configured *event triggers.* Once sudden events, such as the variance of the most recent values is above a threshold, or the number of unreplied requests is considerably high, an update operation is triggered (see tick 40). Then, when the metric values are stable, $\overline{t^s_u}$ is extended by one $\overline{t_s}$ in each update cycle. In our examples $\overline{t_s} = 20$, therefore, after tick 40 the next update intervals have the lengths $\overline{t_s}(= \overline{t^s_{u_{min}}})$, $2 \cdot \overline{t_s}$, and $3 \cdot \overline{t_s}$. However, at tick 180 a sudden behavior change is detected and trust is sampled as soon as possible (instead of waiting a period of $4 \cdot \overline{t_s}$.

Typically some simple and easily computable metrics that characterize the actor behavior and can efficiently capture behavior changes, are used to trigger update actions. While at least this set of metrics, is calculated at each $\overline{t_s}$, the larger amount of (more complex) metrics and finally trust are refreshed only after adaptive $\overline{t^s_u}$. This is visualized in Figure 8(b). Sampled trust $\widehat{\tau}$ is refreshed only at intervals $\overline{t^s_u}$. However, a temporal evaluation (Eq. 5) is still applied at each t_i (as in the fundamental approach), but based on the most recent $\widehat{\tau}$.

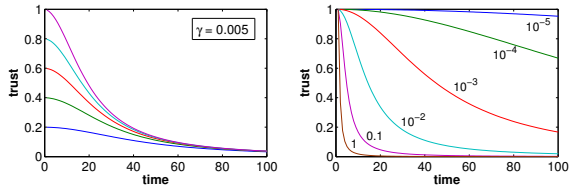
5.2.3. Trust Aging Model

As social relations in the real world degrade if people do not frequently interact, also trust relations underlie an aging process. While it is intuitive that relations will become invalid over time, it is quite hard – if not impossible – to realistically reflect this aspect in a mathematical model. Our approach, as defined in Eq. 7, provides some parameters for tuning the aging process, while it is not too complex to be applied in real environments.

$$\tau^s_n = \tau^s_i \cdot e^{-(\tau^s_{n-1} \cdot \Delta t)^{2\gamma}} \tag{7}$$

The variable τ_i^s represents the latest determined trust value based on interactions in the update procedure that is degraded exponentially, configured by the decay factor γ ($\gamma \geq 1$). So, trust τ_n^s at time tick t_n is calculated by degrading τ_i^s depending on the time span $t_n - t_i$.

The quality of relationships, i.e., trust, suffers if relations are not periodically refreshed through new interactions. While immediately after updating a relation ($\Delta t = 0$), trust is not altered ($\tau_n = \tau_i$), the aging process produces trust results asymptotic to zero with $\Delta t \to \infty$ as shown in Figure 9(a).



(a) Varying levels of trust τ. (b) Varying decay factors γ.

Figure 9: Illustrative example of trust aging from different trust levels τ and for different decay factors γ.

In our model, trust degrades with different speed for different trust levels, i.e., relations between several actors that stopped interacting at the same time are removed simultaneously from the network, independent from their varying levels of trust. The configuration of our aging model is still an open issue. On the one side domain experts could care for this based on best practice, on the other side there exist concepts that let systems adapt parameters autonomously to optimize the aging process. However, we design our trust model to be flexibly enough to cover various demands on temporal properties; e.g., sampling intervals ($\overline{t_s}$), impact of new values (α), decay factor (γ) (see Figure 9(b)).

Adaptive aging refers to the dynamic adaptation of γ, hence, the older a relation, the slower may be the applied aging process. Furthermore, as in real life, the decay of trust can be comparably fast in the beginning, while the actual removal of relations takes longer time. The adaptation of the decay factor may depend on actors interaction consistency (see actor uniformity in the beginning of this section).

5.3. Dynamic Web of Trust Algorithm

Here, we present an algorithm that applies all aforementioned concepts to manage the dynamics in the *Web of Trust* upon observed interactions. This approach accounts for contextual constraints of interactions (scopes), properties and evolutions of trust relations, selective and adaptive update intervals, and the variance of measured metrics. Algorithm 1 demonstrates the adaptation of relations in the Web of Trust.

Algorithm 1 Update of trust graph $G_T = (N, E_T)$ with recently captured interaction logs between $N' \subseteq N$ in $G_I = (N, E_I)$ every discrete tick t_i.

```
1: /* access G_I = (N, E_I) in interaction databases */
2: /* access G_T = (N, E_T) in the trust database */
3: for each n_i ∈ N' do
4:     for each n_j ∈ N' do
5:         for each s ∈ Scopes(edge(E_T, n_i, n_j)) do
6:             if |E_I(n_i, n_j)| > ϑ_I^s then
7:                 /* if enough interactions to reliably infer trust */
8:                 e_τ = E_T(n_i, n_j))
9:                 if ∃ τ^s ∈ e_τ then
10:                    /* update of existing relations scheduled */
11:                    if isUpdateScheduled(e_τ, s) then
12:                        /* previously scheduled update */
13:                        M^s(n_i, n_j) = calcMetrics(E_I(n_i, n_j), s)
14:                        τ̂^s(n_i, n_j) = infTrust(n_i, n_j, M^s(n_i, n_j))
15:                        t̄_u^s = getUpdateInterval(e_τ, s)
16:                        if t̄_u^s ≤ t̄_umax^s then
17:                            scheduleUpdate(e_τ, s, t̄_u^s + t̄_s)
18:                        else
19:                            scheduleUpdate(e_τ, s, t̄_umax^s)
20:                    else
21:                        /* trigger changing behavior */
22:                        M_T^s(n_i, n_j) = calcTriggers(E_I(n_i, n_j), s)
23:                        if isUpdateTriggered(e_τ, M_T^s(n_i, n_j)) then
24:                            M^s(n_i, n_j) = calcMetrics(E_I(n_i, n_j), s)
25:                            τ̂^s(n_i, n_j) = infTrust(n_i, n_j, M^s(n_i, n_j))
26:                            scheduleUpdate(e_τ, s, σ_τ, t̄_umin^s)
27:                        else
28:                            /* stable behavior, no updates */
29:                            τ̂^s(n_i, n_j) = τ̂_{i-1}^s(n_i, n_j)
30:                    /* smoothen trust values */
31:                    τ_i^s(n_i, n_j) = update(τ_{i-1}^s(n_i, n_j), τ̂^s(n_i, n_j))
32:                else
33:                    /* introduce new trust relations */
34:                    M^s(n_i, n_j) = calcMetrics(E_I(n_i, n_j), s)
35:                    τ_i^s(n_i, n_j) = setInitialTrust(M^s(n_i, n_j))
36:                    addTrustRelation(e_τ, s, τ_i^s)
37:                    scheduleUpdate(e_τ, s, t̄_umin^s)
38:            else
39:                /* if too few interactions */
40:                t̄_u^s = getUpdateInterval(e_τ, s)
41:                if t̄_u^s ≤ t̄_umax^s then
42:                    /* increase update intervals */
43:                    scheduleUpdate(e_τ, s, t̄_u^s + t̄_s)
44:                else
45:                    /* age out existing relations */
46:                    applyAging(e_τ, s)
47: /* write back updated G_T */
```

In detail, it models the emergence of new relations (Line 33), updates of existing ones (Line 10), and their aging in case no interactions take place between trusted actors (Line 39).

Our algorithm manages trust between a subset of nodes $N' \subseteq N$ in the *Web of Trust* $G_T = (N, E_T)$ for a predefined set of scopes (depending on already existing relations that need to be updated – see Line 5). In case the amount of interactions to reliably infer behavior is above a predefined threshold (ϑ_I^s depends on the 'usual' amount of interactions in a scope), new relations are introduced and existing ones updated respectively.

New relations are added to G_T if a significant amount of interactions took place between two actors but no trust relations exist (Line 33). The trust level is inferred from measured metrics and updates are scheduled as soon as possible – still accounting for interaction sparsity in the given scope ($\overline{t^s_{u_{min}}}$).

Updates are performed due to two events: (i) an update has been scheduled for a given relation; (ii) a rapid change in an actor's behavior has been triggered and thus, connecting trust relations have to be updated to reflect this change in the model accordingly. In the first case (see Line 11), update cycles are extended up to $\overline{t^s_{u_{max}}}$ for optimization purposes. Hence, for longer stable interaction behavior of actors, update intervals are increased too. However, if considerable sudden changes in behavior are detected (e.g., someone does not reply to requests anymore) (see Line 21), an immediate update is triggered and consecutive updates are performed in shorter intervals until stable behavior is detected again.

If the amount of interactions drops below a given threshold ϑ_I^s, update intervals are increased to collect a sufficient amount for reliably behavior determination. However, if the update interval would become too long ($> \overline{t^s_{u_{max}}}$), the previously described aging process is applied. Function `applyAging()` (Line 46) is implemented as Eq. 7 that continuously degrades trustworthiness, and finally, removes an existing trust relation from the graph model. Algorithm 1 is periodically executed to keep G_T up-to-date. The execution interval needs to be adapted to the inherent dynamics of the environment. Since the algorithm processes interaction logs and relations only for a subset N' of all nodes, computational effort can be distributed over several instances that handle only parts of the whole *Web of Trust* G_T.

6. The Trusted Information Sharing Framework

We describe our *Trusted Information Sharing* framework that has been first introduced in [7] and extend it to meet requirements of the science collaboration scenario discussed in Section 3. We distinguish between two *modes of sharing*: (i) Activity-centric sharing accounts for the currently jointly processed activity of n_i and n_j. Therefore, information is shared to foster ongoing collaborations. (ii) Scope-centric sharing is about information sharing due to trust in a scope, but without accounting for a concrete activity. This kind of sharing is useful to facilitate future collaborations, i.e., the creation of new joint activities.

Besides the modes we distinguish two different *sharing styles*: (i) Active Sharing pushes information to actual or potential collaboration partners (depending on the sharing mode), e.g., a call for paper via notification and announcement services. (ii) Passive Sharing grants access to personal information when requested by other network members, e.g., when the collaboration network is searched for dissemination opportunities. We focus on the latter kind of sharing style that can be understood as a dynamic access control system.

6.1. Sharing Framework

The main components of our framework and their connections are depicted in Figure 10. The backend services comprise one or more *Information Repositories* that hold various kinds of information, encoded in XML and defined by XML Schemes (XSDs). An *Information Catalog* enables users to link information from repositories to sharing scopes. Activities, as introduced in our motivating scenario, are managed by an *Activity Management Service* and act as the glue for multi-dimensional collaboration data (see the context model in Figure 3). Especially trust relations that emerge from interactions between users during the execution of activities, are provided by the *Trust Network Provider*. A *Sharing Rule Management Service* provides trust requirements for information sharing, e.g., a minimum degree of personal trust or reputation, and the *Sharing View Management Service* stores transformation scripts (XSLTs) to customize the view on XML-encoded information.

The *Administration Middleware* is utilized by users to register potentially shared information in the platform. This component provides all features to enable the upload of XML documents, the creation of catalog entries to register the ownership of information and sharing in context of activities, the retrieval of interaction metrics and trust values of relations to information consumers, and the definition of sharing rules and restricted views on the uploaded documents. In the end-user collaboration portal an *Administration Tool* is provided, that communicates with the Administration Middleware. Users can register and unregister information, and create and modify their sharing rules. Furthermore, they have the ability to register new information types (XSDs).

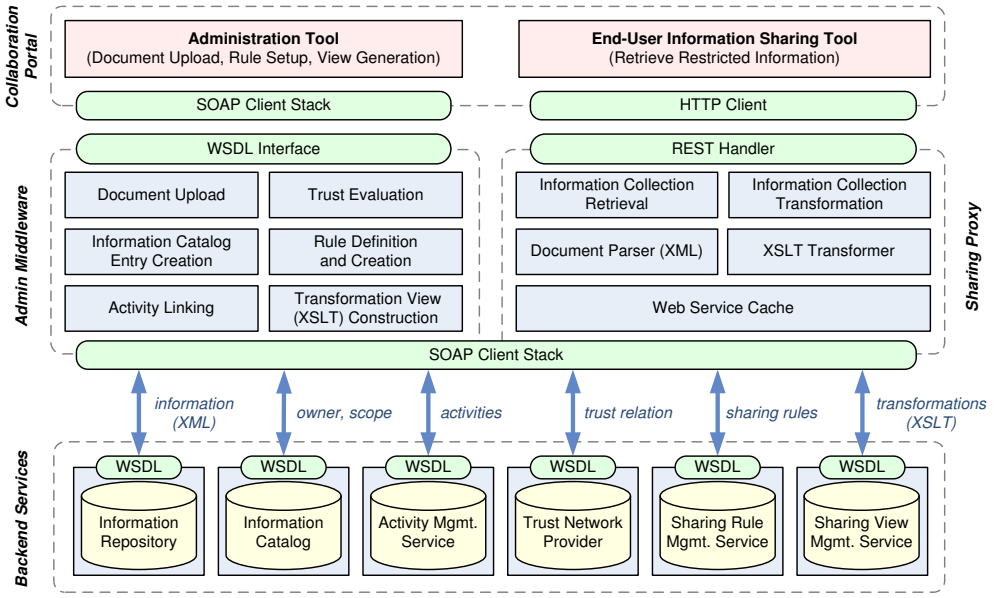
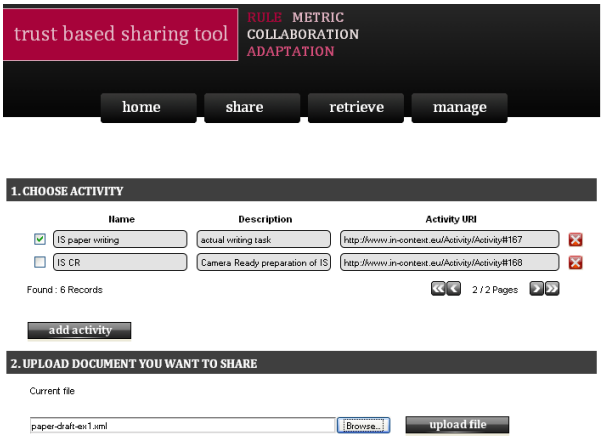


Figure 10: Sharing framework overview.

The *Sharing Proxy* enables users to retrieve information from collaboration partners. This component utilizes all the aforementioned SOAP-based backend services and restricts information based on sharing views picked by evaluating sharing rules. Technically, this is realized by transforming XML data through applying XSLTs depending on trust relations between information owner and requester. Higher trust allows more details to be shared. Since the Sharing Proxy has to serve many concurrent requests and heavily relies on SOAP-based Web services in the backend, we integrated a Web service cache that buffers all responses from the backend. The configuration of cache update intervals is closely linked to the volatility of trust relations and, hence, to update and aging mechanisms discussed before. The *End-User Information Sharing Tool* provides a convenient user interface that enable people to browse the Web of Trust and retrieve information that is shared by collaboration partners. Each user of this tool retrieves his/her individually restricted view on shared information based on their personal relations.

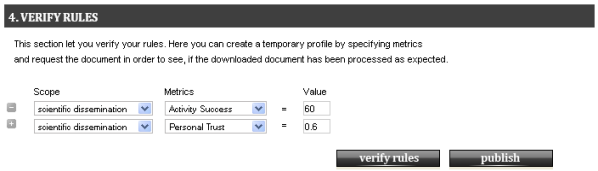
6.2. Administration Tool

Figure 11 shows the end-user's perspective of trusted information sharing. In the first step, as depicted in Figure 11(a) the user picks ongoing activities from a list where s/he wants to publish information. Second, the user uploads the actual document. The document content is modeled as an XML structure and follows a specific schema (XSD). In our example, the user shares a paper draft consisting of title, authors, abstract, keywords, and body, within a dissemination activity. After uploading the document, it is parsed in the administration middleware and all available XML tags are extracted. Then (Figure 11(b)), users are able to define sharing rules on these XML tags. All uploaded information is shown to others by default if no further restrictions are defined. Let us assume for the depicted example that the owner of the paper draft only wants close collaboration partners to see participating authors. Thus, after upload the user restricts access to the author section of the paper draft. A constraint is for instance that a certain requester of the document need to be personally trusted by the document owner with a value equal or higher than 0.5 ($\tau \in [0, 1]$). (Note, values and limits can be set upon best practices or suggestions from domain experts – see [6]). The specification of this rule produces two artifacts: (i) The *sharing view* is an XSLT that transforms the initial paper draft to a version with an omitted authors section. (ii) The *sharing rules* model constraints based on relation metrics for transforming the document. Thus, whenever someone who is not personally trusted with $\tau \geq 0.5$ requests the paper draft, s/he only receives a version without the authors section. Finally, as shown in Figure 11(c), the effects of configured rules can be verified by the document owner. For that purpose, artificial metric values can be specified and the document retrieved in its restricted version for



(a) Selection of activities for potential information sharing and upload of potentially shared information (XML document).

(b) Specification of XML elements to share (=sharing view) and conditions of relation metrics (=sharing rules); Definition of individual views by dynamically creating and parameterizing transformation scripts (XSLTs).



(c) Verification of rules by specifying synthetic relation data (metrics) and final publishing of all required artifacts (information, catalog entry, rules, and view).

Figure 11: Set up sharing of document-based information with trustworthy collaboration partners.

the role of a document consumer, i.e., interested collaboration partner. Publishing the document means that the paper draft is stored in an information repository, a catalog entry is produced that links the document to certain activities, and generated views and rules are deployed in the respective backend services.

6.3. End-User Information Sharing Tool

The Web-based tool for exploring shared information is shown in Figure 12. The user is able to explore his/her network visualized as (undirected) graph. The collaboration network is established based on past interactions as discussed previously. The first view in Figure 12(a) shows a personalized view on the collaboration network (i.e., based on the member with most connections to other members). Users with just one single connection within the network are visualized in a different color. The link weight is proportional to the number of interactions between network members, thus being a direct indicator for the level of trust. As the next step (see Figure 12(b)), a user explores shared information from another member. Again, the link weight and trust restrict how much information is shared between network members. An example for a shared (restricted) information view is shown by Figure 12(c). A detailed explanation on applied rules and transformations is given in the next section. We intended to design a lightweight tool for graph visualization and information sharing. The presented tool has been implemented on top of state-of-art Web toolkits and a JavaScript based visualization toolkit[6]. This has the advantage that collaboration networks can be visualized without requiring additional client side libraries or browser plugins.

7. Design and Implementation

The most basic use case is depicted by Figure 13: A network member n_i (the trustor) has established trust in his activity collaboration partner n_j (the trustee) due to previous cooperative behavior in a specific scope. Therefore, the owner (trustor n_i) of some information, identified by an *uri*, is willed to share this information with his trustee n_j.

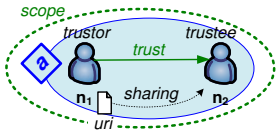
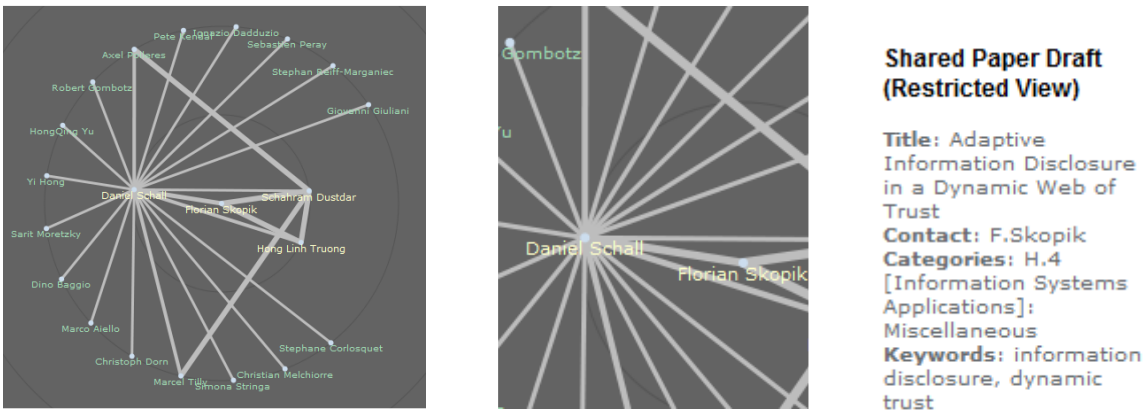


Figure 13: Fundamental sharing scenario.

7.1. Fundamental Mode of Operation

We describe the interplay of the components to cover the fundamental use case of trustworthy sharing of a *particular* information (i.e., that is already referenced by an *uri*), of the owner n_i with the requester n_j. Let us assume, n_i explicitly publishes the *uri* of an XML file in a public area of the collaboration platform. User n_j wants to retrieve this information through the REST interface of the Sharing Proxy, and view in his/her Browser. That is the point, where *trustworthiness* comes

[6]Visualizations for the Web: `http://thejit.org/`



(a) Browse network members. (b) Trusted information discovery. (c) Shared information view.

Figure 12: Browsing shared information in a Web of trust.

into play. The sequential interactions of the single components are depicted in Figure 14. The process starts with retrieving registered meta-data for the given information, including the owner and valid a scope of sharing. After that, joint scopes are requested from the Activity Management Service, i.e., the scopes of all currently running joint activities. Then, the sharing rules of the information owner are retrieved, as well as existing trust relations in the potential sharing scopes. The Sharing Proxy picks the sharing rule that results in the least restrictive information. This means sharing relies on the tightest available trust relation between owner and requester. According to the picked rule, the corresponding XSLT script from the Sharing View Management Service is requested, as well as the initially requested information from the Information Repository. Finally, the initially requested information is transformed to its trustworthy view and delivered to the requester.

7.2. Implementation Details

The information sharing framework, depicted in Figure 10, is designed as distributed service-oriented system. The backend components and the middleware for the administration features are implemented as Axis2 Web services with SOAP interfaces (described with WSDL), and the Sharing Proxy is realized as a Tomcat Servlet with REST access. In this section we highlight design decisions and implementation details.

Sharing Proxy Interface. In contrast to the other components, the Sharing Proxy is not implemented as a SOAP-based Web service, but as a Servlet with a RESTful interface [56]. On the one side, this fact simplifies the integration with the collaboration portal (JSR-168 portlets[7]), on the other side, processing pure HTTP requests deem to be more scalable than SOAP messages. Resource repositories are typical applications for RESTful interfaces, where each resource is explicitly identified by a corresponding uri. The requester is identified by HTTP authorization in each request, therefore no further parameters than the uri of the information of interest is required to enable trusted information sharing. Table 2 summarizes the available RESTful operations of the Sharing Proxy.

Op.	uri/scopeId/activityId/memberId/listInfos&type=xsd
`GET`	get all information uris (collection overview)
`PUT`	–
`POST`	(registration of information through the admin tool)
`DELETE`	–
Op.	**uri/scopeId/activityId/memberId/infoURI**
`GET`	get information identified by uri
`PUT`	update existing information (only with same XSD)
`POST`	–
`DELETE`	delete information (if the requester is the owner)

Table 2: Sharing Proxy REST-style interface.

The uri for each resource is composed of the uri of the proxy servlet with additional scopeId, activityId, memberId, and optional infoURI. If the requester omits the infoURI, a collection of all information (with optional type selection) identified by the given uri is returned (*/listInfos&type=XSD*). Restrictions on scopes, activities, and members are not mandatory, and can be replaced with *anyScope*/*anyActivity*/*anyMember*. For instance, links to all shared paper drafts of any com-

[7] `http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html`

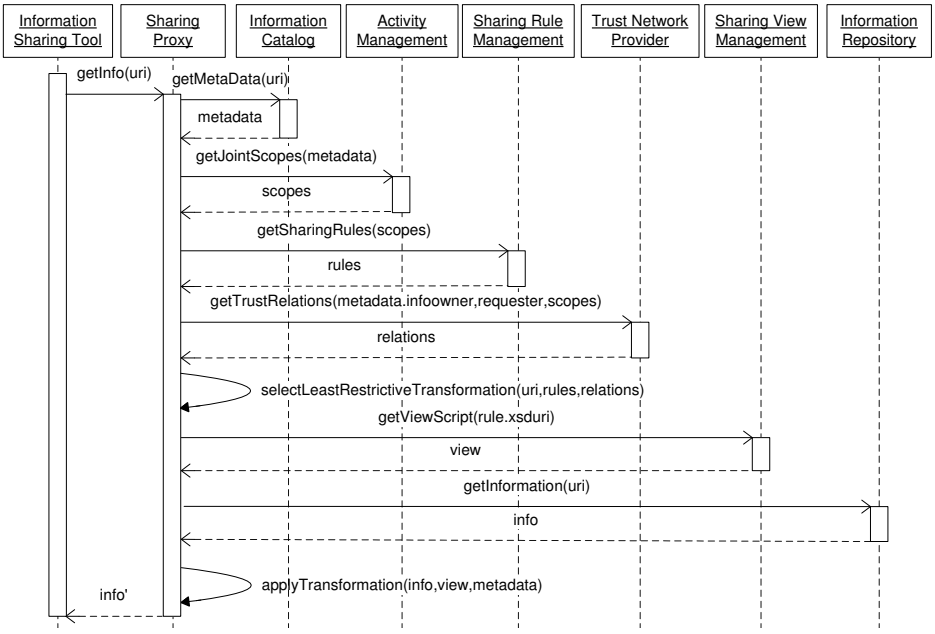


Figure 14: Fundamental interactions of components when retrieving information.

munity member in the scope of 'scientific dissemination', can be found in *uri/disseminationScopeId/anyActivity/anyMember/listInfos&type=paperdraft.xsd*.

Trust Network Provider Interface. Network members retrieve data about connected neighbors in a system-managed trust graph (see Listing 3), and can search for users by name and profile data (similar to a lightweight service registry). Furthermore, the service offers information about someone's trust relations, second-hand recommendations, and third-party reputation.

```
Member[] getTrustors(Uri mTo, Scope s);
Member[] getTrustees(Uri mFrom, Scope s);
Member[] searchMembers(Scope s, Filter f);
double getTrust(Uri mFrom, mTo, Scope s);
double getRecom(Uri mFrom, Uri mTo, Scope s, Filter f);
double getRep(Uri mTo, Scope s, Filter f);
Metric getMetric(Uri mFrom, mTo, Scope s, String name);
Scope[] getAllScopes(Uri mFrom, Uri mTo);
```

Listing 3: Trust provider interface excerpt.

Information Definitions and Repository. Shared information has one of the following origins: (i) information that is manually managed by users, such as documents, notes, and source code in external repositories; and (ii) information that is generated and managed by the system according to the context model. All information structures are pre-defined by XSDs, provided by administrators of the platform. Listing 4 shows exemplarily a paper draft XSD that is suitable for the academic research network scenario and further used in the evaluation part.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:p="http://www.infosys.tuwien.ac.at/tis/papercon"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
 <xsd:import namespace="http://www.infosys.tuwien.ac.at/tis"
  schemaLocation="paperconcepts.xsd"/>
  <xsd:element name="paperdraft" type="tpaperdraft"/>
  <xsd:complexType name="tpaperdraft">
   <xsd:sequence>
    <xsd:element name="title" type="xsd:string"/>
     <xsd:element name="author" type="p:author"
      maxOccurs="unbounded"/>
     <xsd:element name="contact" type="xsd:string"/>
     <xsd:element name="category" type="p:category"
      maxOccurs="unbounded"/>
     <xsd:element name="keywords" type="xsd:string"
      maxOccurs="unbounded"/>
     <xsd:element name="abstract" type="xsd:string"/>
     <xsd:element name="body" type="xsd:string"/>
     <xsd:element name="lastChangeAt" type="xsd:dateTime"/>
     <xsd:element name="linkedRes" type="xsd:anyURI"
      maxOccurs="unbounded"/>
    </xsd:sequence>
   <xsd:attribute name="uri" type="xsd:anyURI"
    use="required"/>
  </xsd:complexType>
</xsd:schema>
```

Listing 4: XSD for information about paper drafts.

Information Registration. Users register each item of information that they intend to share in the Information Catalog (however, this can be automatized with more advanced tool support). By creating catalog entries, they link information (identified by uris of XML data and corresponding XSD(s)) to scopes. In this way, users decide on their own which information can be shared in which scopes. Listing 5 shows an excerpt of the schema of such catalog entries. The main advantage of separating the actual information (e.g., paper drafts)

from sharing management data (e.g., scope of sharing, owner, mode) is that the same information can be linked to different scopes, and links can be dynamically modified without affecting the actual information (*separation of concerns*). The schema (Listing 5) is designed to enable multiple types of search queries, such as retrieving shared information in a scope, of a specific type (XSD), of a particular user, or combinations of these parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
 <xsd:element name="entry" type="tEntry"/>
 <xsd:complexType name="tEntry">
  <xsd:sequence>
   <xsd:element name="registeredName" type="xsd:string"/>
   <xsd:element name="infoXSD" type="xsd:anyURI"/>
   <xsd:element name="infoURI" type="xsd:anyURI"/>
   <xsd:element name="owner" type="xsd:anyURI"/>
   <xsd:element name="scope" type="xsd:anyURI"/>
   <xsd:element name="mode" type="tmode"/>
   <xsd:element name="registeredAt" type="xsd:dateTime"/>
   <xsd:element name="updatedAt" type="xsd:dateTime"/>
   <xsd:element name="comment" type="xsd:string"/>
  </xsd:sequence>
 <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
 </xsd:complexType>
 <!-- tmode omitted -->
</xsd:schema>
```

Listing 5: Catalog entry schema excerpt.

Sharing Rule Definitions. In addition to catalog entries, users who want to share information also define sharing rules that account for dynamically changing trust relations.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"...>
 <xsd:element name="rule" type="tRule"/>
 <xsd:complexType name="tRule">
  <xsd:sequence>
   <xsd:element name="owner" type="xsd:anyURI"/>
   <xsd:element name="validScope" type="xsd:anyURI"/>
   <xsd:element name="applyOnType" type="xsd:anyURI"/>
   <xsd:element name="condition" type="tCondition"/>
   <xsd:element name="applyXSLT" type="xsd:anyURI"/>
   <!-- ... -->
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="tCondition">
  <xsd:sequence>
   <xsd:element name="trust" type="tnValop"/>
   <xsd:element name="recommendation" type="tnValop"/>
   <xsd:element name="reputation" type="tnValop"/>
  </xsd:sequence>
 </xsd:complexType>
 <!-- tnValop omitted -->
</xsd:schema>
```

Listing 6: Sharing rule schema excerpt.

According to the excerpt in Listing 6, users define in which scope(s) a rule is valid, and which type of information (XSD) is concerned. A condition block describes the actual trust requirements for firing a rule, e.g., minimum personal trust, recommendation, and reputation of the requesting community member. The resulting action is a transformation of the desired information (XML) with a pre-defined XSLT script, to filter content and provide restricted views. If sharing rules collide, e.g., there is a rule for all information of a given type, and a second rule that matches the uri of the requested information, the more specific (second) rule is picked.

Sharing View Definitions. Complex XSLT scripts for restricting XML-based information are pre-defined by domain experts (who also define XSDs of information types), and selected by end-users when defining rules.

```
<?xml version="1.0"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/...">
 <xsl:output method="html" encoding="UTF-8" indent="yes" />
 <xsl:template match="/">
  <h3>Shared Paper Draft (Restricted View)</h3>
  <xsl:apply-templates />
 </xsl:template>
 <xsl:template match="paperdraft">
  Title:      <xsl:apply-templates select="title"/> <br/>
  Contact:    <xsl:apply-templates select="contact"/> <br/>
  Categories: <xsl:for-each select="category">
              <xsl:apply-templates/>, </xsl:for-each> <br/>
  Keywords:   <xsl:for-each select="keyword">
              <xsl:apply-templates/>, </xsl:for-each> <br/>
  </xsl:template>
</xsl:transform>
```

Listing 7: Exemplary view on paper drafts.

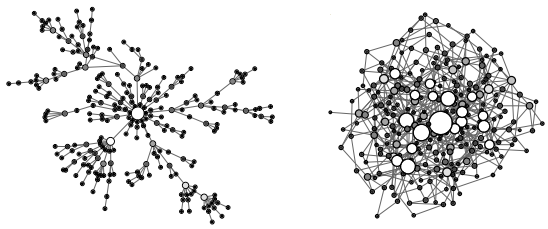
However, for simple document structures, a Web-based editor for end-users is also available (see Section 6 for details on the end-user tool support). For the exemplary paper draft schema in Listing 4, a matching XSLT could have the structure in Listing 7. After applying this script, only paper title, a contact person, categories, and keywords are visible to the requester, while the actual (co-)authors, abstract, document body, modification date, and linked resources are omitted. The output of the transformation process is a HTML fragment that is directly embedded in a dynamic (X)HTML page and rendered in a Portlet of the Collaboration Portal.

8. Evaluation and Discussion

We evaluate the most critical parts of our framework that enable trusted information sharing in a dynamically changing Web of trust. The experiments deal with typical issues, such as the context-aware mining of interaction logs and processing of complex graph structures. In particular, we demonstrate (i) the measurement of interest similarity, (ii) the calculation of support reciprocity, and performance of inferring trust, (iii) demonstrate the advantage of adaptive trust updates, and (iv) measure the end-to-end performance for trusted information sharing in dynamic trust networks.

8.1. Experiment Setup

Collaboration Network Generation. Since we have not yet applied our approach in real large-scale environments, we do not have sufficient real testing data.



(a) average degree d=2. (b) average degree d=4.

Figure 15: Generated scale-free network structures for trust inference performance studies in hierarchical and democratic communities.

Therefore, we use a mix of captured and synthetic data to test and discuss trust emergence, update, and aging mechanisms. We generate artificial scale-free network structures that we would expect to emerge under realistic conditions in science collaboration [50]. For that purpose, we utilize the *preferential attachment model* of Barabasi and Albert [50] to create[8] graphs with power-law distributed degrees depicted[9][10] in Figure 15. These network structures are the basis to generate interaction logs that follow a realistic distribution among members.

In both network structures, nodes follow a power-law distribution. However, in Figure 15(a) we set the average degree $d = 2$ which is typical for hierarchical structures, where most actors receive orders from higher levels and may delegate them to lower levels. In contrast to that, higher average degrees, such as $d = 4$ in Figure 15(b) occur predominantly in more democratic structures, where actors are highly interlinked and may send and receive tasks from virtually anyone. We focus on the latter kind of networks that emerge due to support reciprocity – a property that can only be found in democratic networks – perfectly fitting to academic science collaboration.

Trust Model Setup. As described in Section 4, we infer trust by interpreting various measured metrics. One of these metrics in the science collaboration scenario is interest similarity (*isim*). A second metric describes the interaction reciprocity (*recpr*), i.e., the willingness to support reliable collaboration partners. Changing interaction behavior is triggered by varying availability (*avail*) of actors regarding requests from other members in the network. This means that *avail* is periodically sampled, while trust relations are updated based on *isim* and *recpr* only due to major changes of *avail* (or the maximum update interval has been reached). All three metrics *isim*, *recpr*, and *avail* have been defined in Section 4.

We argue that these metrics appropriately reflect trustworthy behavior in science collaboration. In particular, for successful collaboration mutual interests are of importance, while also a cooperative behavior (expressed by support reciprocity) is highly rewarded. In contrast to science collaboration, in an emergence help and support environment (see [52]) fast and reliable response behavior is of paramount importance; thus, different metrics denote trustworthy behavior there.

Services and Application Hosting. For the following experiments, the interaction logging facilities, metric calculation modules, and trust inference engine (see [7] for details) are hosted on a machine with Intel Xeon 3.2GHz (quad), 10GB RAM, running Tomcat 6 with Axis2 1.4.1 on Ubuntu Linux, and MySQL 5.0 databases. Furthermore, the Sharing Proxy, Administration Middleware, and the backend services are hosted on the same server. The end-user tools, i.e., the Administration Tool and Sharing Tool, are implemented in ASP.NET 3.5 and deployed on an IIS. The client simulation used to produce concurrent document requests, runs on a dedicated Pentium 4 with 2GB RAM on Windows XP, and is connected with the server through a local 100MBit Ethernet.

8.2. Interest Similarity Measurement

As explained in Section 4, the similarity of interests (expressed in profiles) has influences on trust relations between collaboration partners [28, 29, 30, 31]. In contrast to common top-down approaches that apply taxonomies and ontologies to define certain skill-, expertise-, and interest areas, we follow a mining approach that addresses inherent dynamics of flexible collaboration environments. In particular, interests dynamically change over time, but are rarely updated if they are managed manually in a registry. Hence, we determine and update them automatically through mining. For that purpose, we extract keywords of interaction data (see Listing 1) and potential other sources that characterize an actor's center of interests.

The creation of interest profiles upon tagging data has been studied in [29]. That work assumes that users tag resources, such as bookmarks, pictures, videos, articles; and thus express their distinct interests in these objects. In particular, a dataset from `citeulike`[11] expresses people's use and understanding of scientific arti-

[8] JUNG graph library: `http://jung.sourceforge.net`

[9] The node size is proportional to the degree; white nodes represent hub nodes.

[10] Note, here graphs consist only of 250 nodes for better visibility. However, we use graphs with up to 10 000 nodes in our experiments.

[11] `http://www.citeulike.org/`

cles through individually assigned tags, which perfectly matches to our science collaboration scenario.

Table 3: Citeulike tagging data set characteristics.

property name	property value
Number of articles	1020622
Number of articles (tagged by > 50 users)	25
Number of distinct tags	287401
Number of distinct tags (used by > 500 users)	272
Number of distinct users	32449
Average number of tags per article	1.2
Average number of users per article	3.5

We use this data (see Table 3) to create interest profiles bottom-up based on tags (ATPs - actor tagging profiles) and manage them in a vector space model [29]. Whenever actors tag new objects, their ATPs can be updated without requiring manual intervention. However, since arbitrary tags may be freely assigned – there is no agreed taxonomy in `citeulike` – no strict comparison can be performed. Therefore, we cluster tags according to their similarities (measured by the frequency of co-occurrence on the same object) 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 [29] and therefore out of scope of this work. But we outline the evaluation results to demonstrate the applicability of *automatic actor profile creation* and *cluster similarity measurement*, supporting the calculation of interest similarities.

We determine for 25 representative `citeulike` users (users with more than 50 tags distributed over at least 5 articles) their tagging profiles (ATPs). Then we compare these ATPs (300 comparisons) 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 particular, are some actors indeed more similar in terms of tag usage and are they clearly distinguishable from the majority? Figure 16 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 (not shown in Figure 16). As depicted, especially on L3 and L4 a small set of highly similar ATPs are identified, while the majority is still

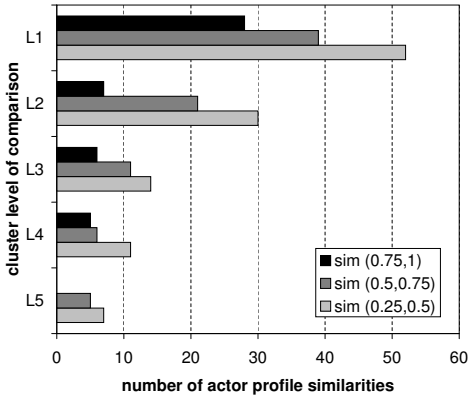


Figure 16: Similarity results among 25 realistic actor tagging profiles (ATPs).

recognized as clearly different. This is the desired effect to distinguish interest similarities between a given user and only a small group of the whole community.

The process of extracting tags and keywords is time-intensive. Besides retrieving the data, some post-processing is required; for instance, removing stop words and ambiguous expressions (e.g., 'imported', 'toread', 'paper') that distort similarity measurements. The overall performance of this process highly depends on the utilized data source (e.g., social platform). However, since interests usually do not change very quickly the determination of *isim* is not time-critical, and therefore, further performance studies are omitted here.

8.3. Collaboration Network Management

The generated graph structures (Figure 15) are the basis for creating realistic interaction logs that are used to conduct some fundamental trust inference performance studies. For a graph $G = (N, E)$, we generate in total $100 \cdot |E|$ interactions between pairs of nodes (n_i, n_j). In our experiments we assume that 80% of interactions take place between 20% of the most active users (reflected by hub nodes with high degree). Generated interactions have a particular type (support request/response, activity success/failure notification) and timestamp, and occur in one of two abstract scopes. While we payed attention on creating a realistic amount and distribution of interactions that are closely bound to node 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). Furthermore, we created required collaboration artifacts, including a pool of activities, some information to be shared, catalog entries, and common sharing rules (accounting for trust

and recomendation) and sharing views.

Through utilizing available interaction properties, we calculate three metrics (i) *availability* (amount of responded support requests), (ii) *interest similarity* (based on extracted tags from successfully finished activities – see Figure 3), and (iii) *support reciprocity* (ratio of served to requested support). Trust in two independent abstract scopes is inferred by applying a fuzzy inference approach (see [6] for details) using the rule base in Listing 2. (Note, *avail* will be required in later experiments.) We measure the required time to completely process the interaction logs, including reading logs from the interaction database (SQL), aggregating logs and calculating metrics, inferring trust, and simple non-adaptive updates in the trust graph (EMA with $\alpha = 0.5$). Compared to previous results in [7], we considerably increased the performance by optimizing accesses to the logging facilities. After each update cycle, generated interactions for the most recent time interval are not used any longer and therefore purged. The length of this interval is fixed, i.e., a single step in the simulation, however in real environments carefully configured due to interaction sparsity in certain scopes. After each update cycle recommendations are calculated on top of the trust graph by computing synthetic transitive relations [7]. Reputation is determined by the average trust values of someone's trustors. The performance results in Listing 4 underline that especially for large-scale networks only a periodic offline calculation is feasible.

Table 4: Trust graph management performance results (in seconds) for scale-free networks with N nodes and average degree $d = 4$.

step in trust inference process	$N = 1\,000$	$N = 10\,000$
retrieve and parse SOAP logs	≈ 250	$\approx 3\,500$
calculate metrics *avail*, *recpr*, *isim*	13	128
interpret trust (*ism*,*recpr*)	7	72
calculate recommendations	5	49
calculate reputation (> 2 inlinks)	< 1	< 1

8.4. Effectiveness of Adaptive Update Strategies

We underline the advantages of selective and adaptive updates with several evaluation results. For the following experiments, we set up a simulation environment as follows: In contrast to the previous section, where we measured the performance of the fundamental trust inference approach upon synthetic interactions, we directly model different user behavior here to demonstrate the applicability of adaptive update intervals. In this round-based simulation the metrics *avail*, *recpr*, and *isim* are modified for a fixed amount of actors. In particular, 5%, 10%, and 20% of (erratic) actors change in each round (with length $\overline{t_s}$) these metrics randomly between 1% and 50%. We introduce $G_R = (N, E_R)$ which is a graph reflecting the reality, and modify metrics assigned to its edges $e_r \in E_R$. The trust model is managed in $G_T = (N, E_T)$ and its edges $e_\tau \in E_T$ updated by Algorihtm 1 according to changing metrics in G_R.

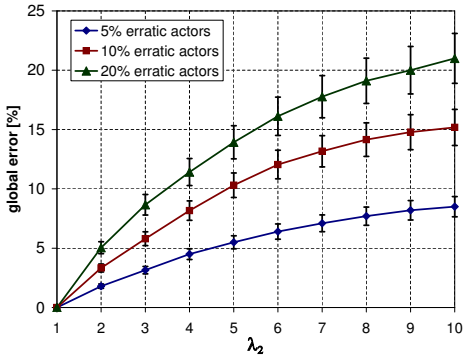


Figure 17: Deviation of trust values (global error) between simulated network and captured model for differently configured update strategies ($\lambda_1 = 1$).

The main goal of adaptive updates, compared to periodic intervals, is the reduction of update cycles due to performance reasons. However, by delaying updates a deviation (Eq. 8) between G_T and G_R is introduced that has to be kept to a minimum. The average deviation $dev(G_R, G_T)$ reflects the effectiveness of update models.

$$dev(G_R, G_T) = \frac{\sum_{e \in E} |\tau(e_r) - \tau(e_\tau)|}{|E|} \tag{8}$$

The proposed update approach in Section 5 has several tuning parameters. Among the most important ones are the settings of minimum and maximum update intervals, configured as $\overline{t^s_{u_{min}}} \leq \overline{t^s_u} \leq \overline{t^s_{u_{max}}}$; whereas $\overline{t^s_{u_{min}}} = \lambda_1 \cdot \overline{t_s}$ and $\overline{t^s_{u_{max}}} = \lambda_2 \cdot \overline{t_s}$ and $\lambda_1 \leq \lambda_2$. Hence, λ_2 allows to extend the scheduled updates of stable relations up to $\overline{t^s_{u_{max}}}$ and thus, to significantly reduce computational effort.

The introduced *global error* due to adaptive updates (compared to fixed interval updates) is expressed as the average $dev(G_R, G_T)$ in percent. Figure 17 depicts this error for different λ_2. In this experiment, the behavior trigger mechanism (compare Line 21 in Algorithm 1) has been deactivated. Instead, we decrease $\overline{t^s_u}$ by one $\overline{t_s}$ after each update operation. Hence, the lengths of future update intervals directly depend on the lengths of recent update intervals, but are only moderately influenced by sudden behavior changes. It is demonstrated that even for small λ_2 considerable error rates are introduced. Since simulated behavior relies on various randomly changed metrics, error bars indicate the spread of

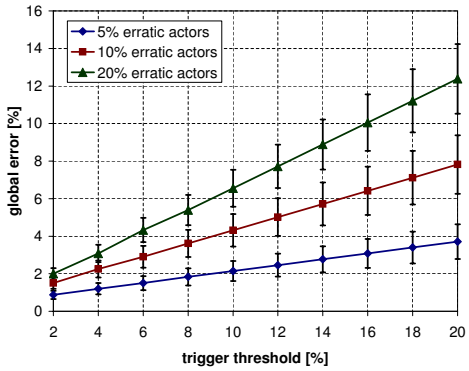


Figure 18: Deviation of trust values (global error) between simulated network and captured model for differently configured trigger thresholds ϑ_t ($\lambda_1 = 1$, $\lambda_2 = 5$).

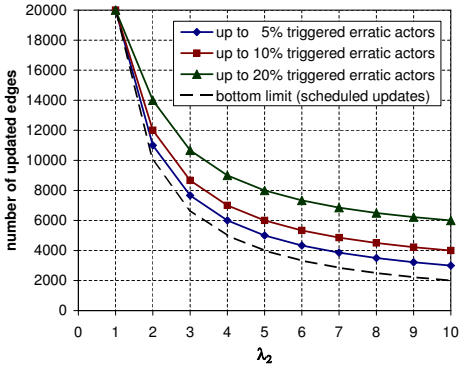


Figure 19: Number of processed edges after updating the trust model according to the simulated network ($|E| = 20\,000$, $\lambda_1 = 1$).

results for multiple runs of this experiment.

Although λ_1 determines $\overline{t^s_{u_{min}}}$, there is also an additional trigger mechanism that initiates immediate updates independent from $\overline{t^s_{u_{min}}}$ if actors change their behavior very quickly. With the trigger threshold ϑ_t the limit of tolerated behavior change without triggering an immediate update can be set. This threshold is defined as the deviation in percent of metric values in the interval $\overline{t_s}$. We trigger behavior changes by frequently observing the metric *avail*. With this trigger mechanisms, an upper limit of global error rate can be guaranteed, because rapid behavior changes (reflected in G_R) are detected and immediate updates of G_T performed. Hence, deviations are not added up over multiple sampling intervals (up to $\overline{t^s_{u_{max}}}$).

Figure 18 visualizes that with the trigger mechanism in place, the global error rates can be considerably decreased. Typically, a higher number of erratic actors in the network still causes a higher average global error. The reason for that is a significant amount of actors who change their behavior slightly below the trigger threshold. Thus, a deviation of G_R to G_T is caused, but no updates triggered. However, setting a smaller λ_2 results in a smaller $\overline{t^s_{u_{max}}}$ and forces frequent updates; therefore, introduces an upper limit of global error rates over time.

Since we have now demonstrated that we can keep the global error rate low, even when we apply adaptive updates (especially with a behavior change trigger in place), we demonstrate now the performance advantages. For that purpose, we utilize a generated graph G_R with 10 000 nodes and 20 000 edges (i.e., $d = 4$). In particular, we investigate the average amount of update operations per $\overline{t_s}$ for differently configured λ_2. Higher λ_2 cause less frequent updates of relations. Note, trust updates of relations are not synchronous, i.e., all at the same point in time, but time instants are set for each edge individually in multiples of $\overline{t_s}$.

Theoretically, without adaptive updates and behavior triggers (i.e., $\lambda_1 = \lambda_2 = 1$), approximately 20 000 ($= |E|$) operations per $\overline{t_s}$ would be required to keep the example graph up to date with an error rate virtually equal to zero. However, since updates may be postponed until $\overline{t^s_{u_{max}}}$ if no rapid behavior changes are detected, the number of required update operations in G_T drops exponentially for higher λ_2, as shown in Figure 19. The dashed line visualizes the number of updated edges due to scheduled updates, even if actors do not change their behavior (then, all updates are performed in intervals of $\overline{t^s_{u_{max}}}$). The other lines show the upper limit of performed updates, in the case that the set of relations with scheduled updates and relations with detected behavior changes do not overlap. Usually, the number of required updates is somewhere between these two limits.

Table 5: Efficiency and effectiveness of adaptive trust updates ($\lambda_1 = 1$, $\vartheta_t = 10\%, amount\ of\ erratic\ actors = 10\%$).

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

Finally, Table 5 summarizes previous results by comparing introduced global errors and required update operations; thus, demonstrating potential savings.

8.5. End-to-End Information Sharing Performance

The overall process of trusted information sharing involves several backend services. Communicating with and retrieving data from these Web services is time-intensive, especially if they are frequently utilized

and/or large amounts of data are transferred (and processed by respective SOAP stacks). Besides the actual *Information Repository*, we identified the *Information Catalog*, *Sharing View Management Service* and *Sharing Rule Management Service* as the most data-intensive services. Therefore, we studied the overall performance when caching different kinds of data. In particular, the *Sharing Proxy* implements the caching strategy of self-pruning cache objects as widely adopted [57].

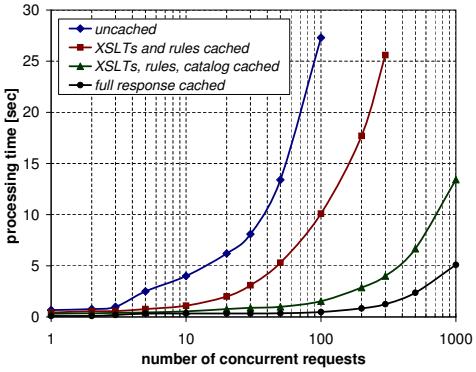


Figure 20: Overall performance of the sharing framework.

Figure 20 depicts the required time of the *Sharing Proxy* to process different amounts of concurrent client requests. In detail, we measured the processing time (i) without any caching mechanisms, (ii) when caching only rarely changed sharing rules and associated sharing views (XSLTs), (iii) when caching rules, XSLTs, and catalog entries, (iv) for delivering the response only, i.e., providing the already transformed and cached information. The results show that with applying different caching strategies the overall performance can be significantly increased. However, depending on the domain's inherent trust dynamics, a trade-off between performance and up-to-dateness of cached data has to be carefully considered.

9. Conclusion and Further Work

In this paper we highlighted the application of the widely adopted MAPE approach for adaptations in complex interaction networks. Adaptation techniques, accounting for contextual constraints and emerging social relations, such as trust, are among the key research areas in flexible service-oriented collaboration environments. Instead of a purely conceptual and algorithmic perspective, we demonstrated the technical grounding, using state-of-the-art Web services technologies. We discussed the realization of a framework that supports a concrete use case, i.e., adaptive information disclosure in large-scale research communities. The evaluation of the running framework discovered important design issues, such as the configuration of dynamic trust models and the need for appropriate caching strategies depending on the scale of supported networks. Our approach has important implications on adaptations in complex systems, because it reduces configuration burdens for the users and permits self-regulation of shared information based on collaboration strength. Our future work includes the deployment and evaluation of the implemented framework in the EU FP7 project COIN. 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 will discover the usability of trust-based adaptive information disclosure in real business environments.

Acknowledgment

The research leading to these results has received funding from the European Union through the FP7-216256 project COIN. We would further like to express our gratitude to Iwona Les for her contribution to the implementation of described concepts.

References

[1] J. Kleinberg, The human texture of information, http://www.edge.org/q2010/q10_index.html (2010).

[2] B. Dybwad, Think twice: That facebook update could get you robbed, http://mashable.com/2009/08/27/facebook-burglary/ (August 2009).

[3] R. Kilner, Internet shopping for burglars on social networks, http://www.insurancedaily.co.uk/2009/08/28/internet-shopping-for-burglars-on-social-networks/ (August 2009).

[4] F. Skopik, D. Schall, S. Dustdar, The cycle of trust in mixed service-oriented systems, in: Euromicro Conference on Software Engineering and Advanced Applications, 2009, pp. 72–79.

[5] F. Skopik, H.-L. Truong, S. Dustdar, Trust and reputation mining in professional virtual communities, in: International Conference on Web Engineering, 2009, pp. 76–90.

[6] F. Skopik, D. Schall, S. Dustdar, Trustworthy interaction balancing in mixed service-oriented systems, in: ACM Symposium on Applied Computing, 2010, pp. 801–808.

[7] F. Skopik, D. Schall, S. Dustdar, Trust-based adaptation in complex service-oriented systems, in: International Conference on Engineering of Complex Computer Systems, 2010, pp. 31–40.

[8] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: International Symposium on Handheld and Ubiquitous Computing, 1999, pp. 304–307.

[9] M. Baldauf, S. Dustdar, F. Rosenberg, A survey on context-aware systems, International Journal of Ad Hoc and Ubiquitous Computing 2 (4) (2007) 263–277.

[10] S. W. Loke, Context-aware artifacts: Two development approaches, IEEE Pervasive Computing 5 (2) (2006) 48–53.

[11] Y. Yang, F. Mahon, M. H. Williams, T. Pfeifer, Context-aware dynamic personalised service re-composition in a pervasive service environment, in: UIC, 2006, pp. 724–735.

[12] B. L. Harrison, A. Cozzi, T. P. Moran, Roles and relationships for unified activity management, in: International Conference on Supporting Group Work, 2005, pp. 236–245.
[13] 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.
[14] 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.
[15] 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.
[16] D. Schall, C. Dorn, S. Dustdar, I. Dadduzio, Viecar - enabling self-adaptive collaboration services, in: Euromicro Conference on Software Engineering and Advanced Applications, 2008, pp. 285–292.
[17] 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.
[18] 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.
[19] J. Breslin, A. Passant, S. Decker, Social web applications in enterprise, The Social Semantic Web 48 (2009) 251–267.
[20] D. Schall, H.-L. Truong, S. Dustdar, Unifying human and software services in web-scale collaborations, IEEE Internet Computing 12 (3) (2008) 62–68.
[21] D. Schall, Human interactions in mixed systems - architecture, protocols, and algorithms, Ph.D. thesis, Vienna University of Technology (2009).
[22] C. Petrie, Plenty of room outside the firm, IEEE Internet Computing 14 (2010) 92–96.
[23] S. P. Marsh, Formalising trust as a computational concept, Ph.D. thesis, University of Stirling (April 1994).
[24] D. Artz, Y. Gil, A survey of trust in computer science and the semantic web, Journal on Web Semantics 5 (2) (2007) 58–71.
[25] T. Grandison, M. Sloman, A survey of trust in internet applications, IEEE Communications Surveys and Tutorials, 2000, 3 (4).
[26] 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.
[27] L. Mui, M. Mohtashemi, A. Halberstadt, A computational model of trust and reputation for e-businesses, in: Hawaii International Conferences on System Sciences, 2002, p. 188.
[28] C.-N. Ziegler, J. Golbeck, Investigating interactions of trust and interest similarity, Decision Support Systems 43 (2) (2007) 460–475.
[29] F. Skopik, D. Schall, S. Dustdar, Start trusting strangers? bootstrapping and prediction of trust, in: International Conference on Web Information Systems Engineering, 2009, pp. 275–289.
[30] J. Golbeck, Trust and nuanced profile similarity in online social networks, ACM Transactions on the Web 3 (4) (2009) 1–33.
[31] Y. Matsuo, H. Yamamoto, Community gravity: Measuring bidirectional effects by trust and rating on online social networks, in: International World Wide Web Conference, 2009, pp. 751–760.
[32] F. Kerschbaum, J. Haller, Y. Karabulut, P. Robinson, Pathtrust: A trust-based reputation service for virtual organization formation, in: International Conference on Trust Management, 2006, pp. 193–205.
[33] 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.
[34] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: International World Wide Web Conference, 2004, pp. 403–412.
[35] P. Massa, P. Avesani, Trust-aware collaborative filtering for recommender systems, in: CoopIS, DOA, ODBASE, 2004, pp. 492–508.
[36] 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.
[37] C.-N. Ziegler, G. Lausen, Propagation models for trust and distrust in social networks, Information Systems Frontiers 7 (4-5) (2005) 337–358.
[38] 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, 2007.
[39] M. J. Metzger, Privacy, trust, and disclosure: Exploring barriers to electronic commerce, Journal on Computer-Mediated Communication, 2004, 9 (4).
[40] S. Marsh, Information sharing is enhanced using trust models, Pervasive Adaptation, 2008.
[41] IBM, An architectural blueprint for autonomic computing, Whitepaper, 2005.
[42] E. D. Nitto, C. Ghezzi, A. Metzger, M. P. Papazoglou, K. Pohl, A journey to highly dynamic, self-adaptive service-based applications, Autom. Softw. Eng. 15 (3-4) (2008) 313–341.
[43] V. Bryl, P. Giorgini, Self-configuring socio-technical systems: Redesign at runtime, International Transactions on Systems Science and Applications 2 (1) (2006) 31–40.
[44] J. Zhang, R. J. Figueiredo, Autonomic feature selection for application classification, in: International Conference on Autonomic Computing, IEEE, 2006, pp. 43–52.
[45] D. Kovac, D. Trcek, Qualitative trust modeling in soa, Journal of Systems Architecture 55 (4) (2009) 255–263.
[46] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, K. Nahrstedt, A trust management framework for service-oriented environments, in: International World Wide Web Conference, 2009, pp. 891–900.
[47] Z. Malik, A. Bouguettaya, Reputation bootstrapping for trust establishment among web services, IEEE Internet Computing 13 (1) (2009) 40–47.
[48] M. Altinel, M. J. Franklin, Efficient filtering of xml documents for selective dissemination of information, in: International Conference on Very Large Data Bases, 2000, pp. 53–64.
[49] Y. Diao, S. Rizvi, M. J. Franklin, Towards an internet-scale xml dissemination service, in: International Conference on Very Large Data Bases, 2004, pp. 612–623.
[50] A. Reka, Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (2002) 47–97.
[51] A. Agrawal et al., Ws-bpel extension for people (bpel4people), version 1.0 (2007).
[52] F. Skopik, D. Schall, S. Dustdar, Trusted interaction patterns in large-scale enterprise service networks, in: Euromicro International Conference on Parallel, Distributed and Network-Based Computing, 2010, pp. 367–374.
[53] D. J. Watts, Six degrees: The science of a connected age, WW Norton & Company, 2003.
[54] B. E. Brewington, G. Cybenko, How dynamic is the web?, Computer Networks 33 (1-6) (2000) 257–276.
[55] C. Domingo, R. Gavaldà, O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, Data Min. Knowl. Discov. 6 (2) (2002) 131–152.
[56] R. T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine (2000).
[57] B. D. Goodman, Accelerate your web services with caching, IBM Advanced Internet Technology, 2002.