Trust-based Adaptation in Complex Service-oriented Systems

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

***Abstract*—Complex networks consisting of humans and software services, such as Web-based social and collaborative environments, typically require flexible and context-based interaction models. Due to the dynamics in such systems, networks are in a state of constant flux and change. Several fundamental concepts, including discovery, interactions, task delegations and executions are no longer based on static policies, but need periodic readjustments. Sophisticated adaptation techniques for improving collaborations are within the key research areas in service-oriented systems. In this paper, we introduce an adaptation approach that accounts for emerging trust relations based on varying interaction behavior of network members. We describe a science collaboration scenario that applies adaptive information sharing techniques. In our model, trust evolves from cooperative behavior of collaboration partners. This behavioral trust provides an intuitive grounding for adaptations and optimizations 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. Furthermore, we present a reference architecture, and evaluate its applicability for large-scale collaboration networks.**

***Keywords*-trust-based adaptation, complex mixed systems, trust emergence, trust-based information views;**

I. Introduction

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

A mixed service-oriented system is a complex network comprising human- and software services that can be flexibly composed to perform various kinds of activities. Therefore, interactions in such systems do not only span humans, but also software services. Recently, *trust* has been identified as a beneficial concept to deal with the dynamics in large-scale networks [1], [2], for example, trust-based selection of services. In contrast to the traditional security perspective on trust, in this context, trust is related to how much humans or other systems can rely on services to accomplish their tasks [3]. Therefore, trust emerging in specific scopes, relies on previous interactions, and evolves over time. Accounting for dynamically changing trust relations when selecting people for communication or collaboration, services to be utilized, and resources to be applied, leads to more efficient cooperation and compositions of human- and software services [4]. Some typical aspects that require run-time adaptation in mixed service-oriented systems 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. In contrast, utilizing periodically updated trust relations better accounts for varying user preferences and avoids lookup based on stale information.
- *Access to and Sharing of Information.* Traditional approaches to access rights management are based on manually assigned static user roles. However, the user is often not able to keep track of configurations in complex networks 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 run-time behavior.
- *Interaction Policies and Patterns.* In common enterprise networks, 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 introduce a general approach to deal with these concerns, enabling trust-based adaptation of complex network structures (Figure 1). This concept follows an adopted version of the 'MAPE' cycle applied in the autonomic computing domain [5]. MAPE, incorporating fundamental concepts from control engineering, describes a cycle consisting 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,

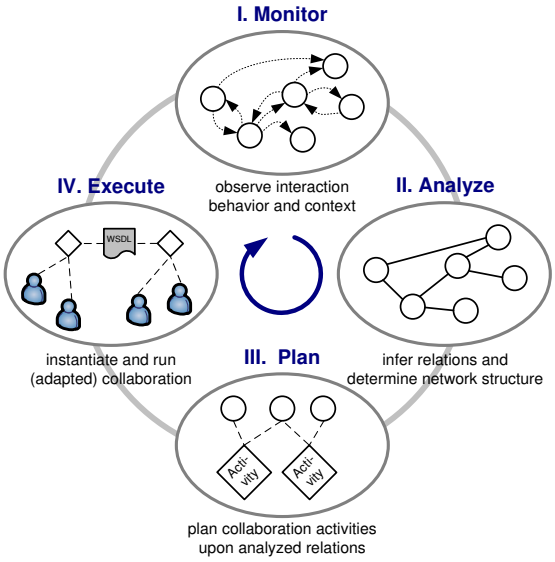


Figure 1. Adaptation approach for mixed service-oriented systems.

therefore, allows to adapt to varying circumstances. This aspect enables us to address aforementioned challenges. In the *Monitoring Phase* our system observes interactions, in particular communication, coordination, and execution events among network members in their respective situations. In the *Analyzing Phase* logged interactions are used to infer relations and to determine network structures. For this purpose, domain-dependent interaction metrics and quality criteria are calculated and interpreted. The following *Planning Phase* covers the preparation of new collaborations, for instance, discovery, ranking and selection of network members, services, and resources. In the *Execution Phase* either new collaborations are instantiated, or existing scenarios adapted according to feedback from prior observations. In that phase network members interact to perform planned activities. This closes the cycle.

Previously, we introduced methods and algorithms that are applied in the monitoring and analyzing phases of the MAPE approach for inferring trust by interpreting and weighting interactions [4]. In this paper, we describe the realization and major design decisions of frameworks that support adaptations in complex service-oriented networks. We present the following contributions:

- *Trust-based Adaptation in Complex Systems.* We focus on complex networks of human and service actors. In that environment, we describe the emergence of trust upon interactions, and discuss a self-adaptive approach as well as typical concerns.
- *Realization and Implementation Aspects.* We discuss the support and realization of one representative mixed complex network example. In that use case, information sharing among network members is adapted by accounting for dynamically emerging trust relations.
- *Evaluation and Discussion.* We evaluate our Web services-based implementation with performance studies under realistic conditions.

The rest of the paper is organized as follows. In Section II we describe a detailed science collaboration scenario, where information sharing is adapted based on varying trust relations. Section III deals with the applied trust model. The following Section IV focuses on the realization of our approach, including architectural and implementation details. We evaluate and discuss our work in Section V. Related work is listed in Section VI, and Section VII concludes the paper.

II. 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 [6] to embed humans acting as services using SOA concepts. This includes WSDL descriptions of interfaces, central registries, SOAP-based interactions, and sophisticated logging facilities.

A. Emerging Trust Networks

Recently, we demonstrated the (semi-)automatic flexible determination of trust [4] in the above-mentioned service-oriented collaboration environment. Briefly, our approach relies on the observation of fundamental interactions, such as SOAP-based communication, coordination or execution

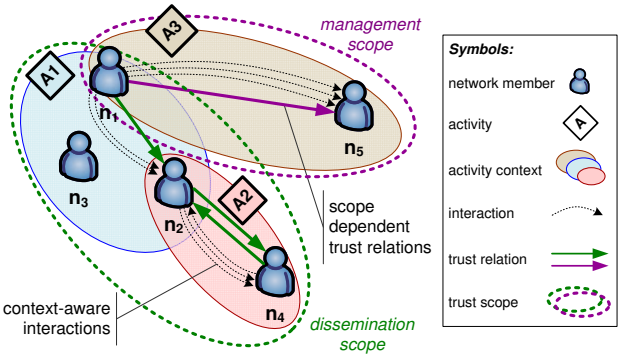


Figure 2. On the emergence of trust.

messages. People interact and use services when conducting activities. Figure 2 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 [4]. 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 2, 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 $A1$ and $A2$, leads to the establishment of trust in one uniform 'dissemination scope'. Finally, a scale-free complex network emerges from cooperations in typical research collaborations as investigated by [7].

B. 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 3, 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

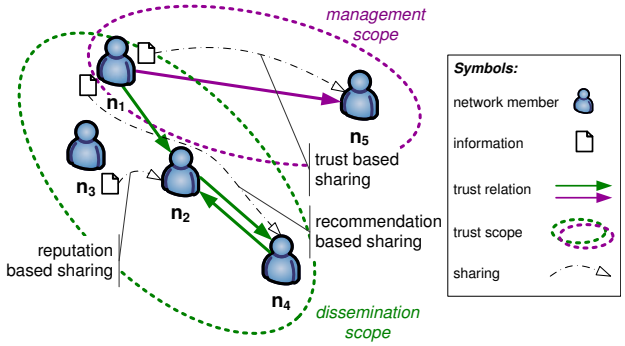


Figure 3. Trust concepts utilized for trusted information sharing.

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 3 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 [8]). 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 [9].

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).

III. The Trust Model in a Nutshell

In virtual communities, where people dynamically interact to perform activities, reliable and dependable behavior promotes the emergence of trust. As collaborations are increasingly performed online, supported by service-oriented technologies, such as communication-, coordination-, and resource management services, interactions have become observable. By monitoring and analyzing interactions, trust can be automatically inferred. In contrast to manual rating approaches, automatic inference is well-suited for complex networks, where potentially thousands of network members dynamically interact. We demonstrated the automatic inference of trust earlier [4], [10], therefore, we highlight the fundamental concept only.

A. Context Model and Shared Information

A model as depicted in Figure 4 is needed to capture the context of interactions, and to distinguish and categorize interaction behavior with respect to different situations. It reflects the relationships between managed information in social and collaborative networks, including the introduced science collaboration network scenario. Briefly, this model comprises the aggregated information from various supporting services, such as community member profiles, calculated interaction and trust metrics, personal information, and involved activities.

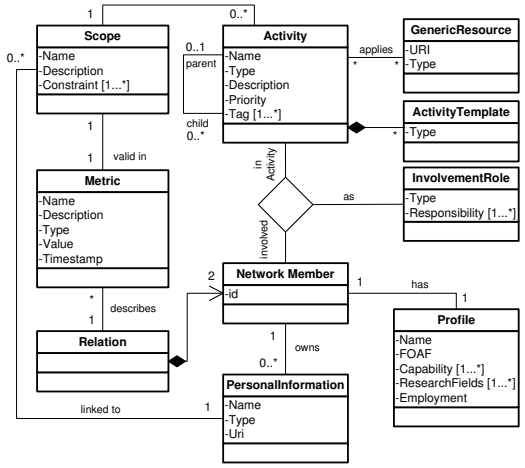


Figure 4. Interaction context model and shared information.

We enable the sharing of all of this information. Hence, in contrast to traditional approaches, such as P2P file sharing that focuses on sharing of document-based information only, we also allow sharing of social information. Besides personal data, this includes profiles, member relationships, activity involvements, regular co-workers, or collaboration performance determined by previous interactions.

B. Fundamental Trust Model

Not only service interactions, but also human interactions may rely on SOAP (e.g., see Human-Provided Services [6] and BPEL4People [11]), which is the state-of-the-art technology in service-oriented environments, and well supported by various 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 with respect to specific situations. We apply an arithmetic calculation of trust based on these metrics. This calculation is context dependent, so in different domains and use cases the impact of metrics varies. As interaction behavior changes over time, trust will alter too. Therefore, trust deems to be an intuitive grounding for flexible adaptation techniques in mixed service-oriented systems.

The interaction behavior of a network member n_i toward n_j is described by various metrics $M(n_i, n_j)$, such as average response time, availability, and rate of joint successful activities. These 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. The sum of selected metrics build the **confidence** $c^s(n_i, n_j) \in [0, 1]$ of n_i in n_j in scope s. This confidence represents recent evidence that an actor behaves dependably, securely and reliably. In Equation 1 confidence is calculated from metrics $m_k \in M(n_i, n_j)$ that are weighted by w_k ($\sum_k w_k = 1$).

$$c^s(n_i, n_j) = \sum_{\forall m_k} (m_k(n_i, n_j) \cdot w_k) \tag{1}$$

The **reliability of confidence** $\rho(c^s(n_i, n_j)) \in [0, 1]$, ranging from totally uncertain to fully confirmed, depends mainly on the amount of data used to calculate confidence (more data provide higher evidence), and the variance of metric values collected over time (e.g., stable interaction behavior is more trustworthy). The function Ψ^s_ρ (Equation 2) determines the reliability ρ of the confidence value $c^s(n_i, n_j)$ relying on utilized metrics $M(n_i, n_j)$. The specific implementation is out of scope of this paper.

$$\rho(c^s(n_i, n_j)) = \Psi^s_\rho(n_i, M(n_i, n_j), s) \tag{2}$$

We infer **personal trust** $\tau^s(n_i, n_j) \in [0, 1]$ by combining confidence with its reliability (see operator $\otimes$ in Equation 3). This can be performed either rule-based by attenuating confidence respecting reliability, or arithmetically, for instance by multiplying confidence with reliability (as both are scaled to the interval $[0, 1]$). Trust is managed in a directed graph $G = (N, E)$.

$$\tau^s(n_i, n_j) = \langle c^s(n_i, n_j), \rho(c^s(n_i, n_j)), \otimes \rangle \tag{3}$$

Recommendation $\tau^s_{rec}(n_i, n_j)$ is built by aggregating n_i's trustees' trust relations to n_j. Hence, recommendation represents second-hand experiences. Potential recommender of n_i for n_j are all $Rec_{ij} \subseteq \{n \in N | \tau^s(n_i, n) \neq \perp \wedge \tau^s(n, n_j) \neq \perp\}$ (The symbol $\perp$ denotes that no relation exists). As common in other models [12] we weight the recommendation of each $n \in Rec_{ij}$ with the trustworthiness of n_i in n (Equation 4).

$$\tau^s_{rec}(n_i, n_j) = \frac{\sum_{n \in Rec_{ij}} \tau^s(n, n_j) \cdot \tau^s(n_i, n)}{\sum_{n \in Rec_{ij}} \tau^s(n_i, n)} \quad (4)$$

Reputation τ_{rep} is similar to recommendation, however, the actor n_i inferring the reputation of n_j does not require a personal trust relation to n_j's trustors ('reputing' entities $Rep_j \subseteq \{n \in N | \tau^s(n, n_j) \neq \perp\}$). Reputation represents a kind of global (community) trust, calculated on top of each trustor's personal trust relations. As in other models, single trust relations can be weighted before aggregation (Equation 5). More advanced models, such as TrustRank [13], may account for the reputation of the trustors too.

$$\tau^s_{rep}(n_j) = \frac{\sum_{n \in Rep_j} \tau^s(n, n_j)}{|Rep_j|} \quad (5)$$

C. Temporal Evaluation

Personal trust $\tau^s(n_i, n_j)$ 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 evaluated at time step i as τ^s_i. As trust is evolving over time, we do not simply replace old values with newer ones, but merge them accordingly. For this purpose we apply the concept of exponential moving average (EMA)[1], to smoothen the sequence of calculated trust values as shown in Equation 6. With this method, we are able to adjust the importance of trust τ^s that is based on the most recent interaction behavior, compared to history trust values τ^s_{i-1} (smoothing factor $\alpha \in [0, 1]$). In case there are no interactions between two entities, but an existing trust relation, the reliability of confidence is lowered by a small amount each evaluation interval. Therefore, trust between entities is reduced stepwise, if they do not interact frequently.

$$\tau^s_i = \alpha \cdot \tau^s + (1 - \alpha) \cdot \tau^s_{i-1} \quad (6)$$

IV. Design and Architecture

The most fundamental use case of trusted information sharing is as follows: A network member n_i (the trustor) has established trust in his 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 a uri, is willed to share this information with his trustee n_j.

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

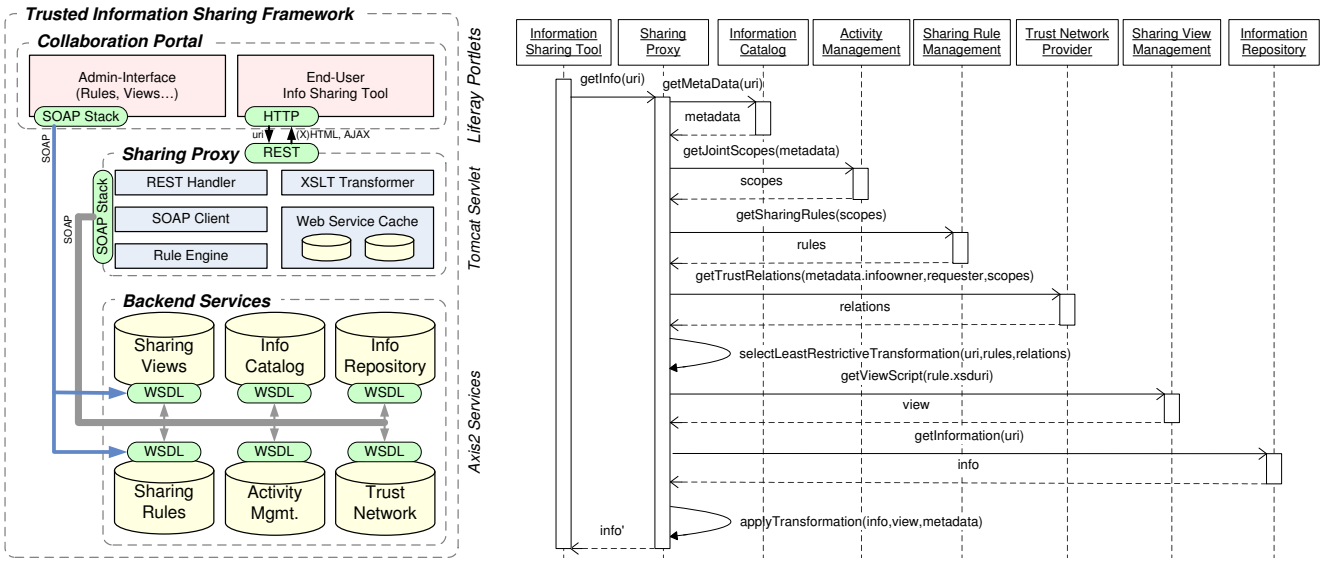
We distinguish between two *modes of sharing*: (i) Activity-centric sharing accounts for the currently jointly processed activity of n_i and n_j. Therefore, information is shared to foster ongoing collaborations. (ii) Scope-centric sharing is about information sharing due to trust in a scope, but without 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 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.

A. Sharing Framework

Structural View. The major components of our framework and their connections are briefly shown in Figure 5(a). 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 4). 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* stores transformation scripts (XSLTs) to customize the view on XML-encoded information. The *Sharing Proxy* utilizes all the aforementioned SOAP-based 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. In the end-user collaboration portal an *Information Sharing Tool* is provided, that communicates with the Sharing Proxy via a REST-style interface, and allows to create, read, update and delete (CRUD) shared information. This includes adding new information to repositories (e.g., document stores) and registering this information in the Information Catalog. An *Administrator Interface* enables the configuration of sharing rules and views (XSLTs), as well as the registration of new information types (XSDs).

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.,



(a) Sharing framework overview.

(b) Fundamental interactions of components when retrieving information.

Figure 5. Architecture and component interactions in trusted information sharing.

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 Browser. That is the point, where *trustworthiness* comes into play. The sequential interactions of the single components are depicted in Figure 5(b). The process starts with retrieving registered meta-data for the given information, including the owner and valid scopes of sharing. After that, joint scopes are requested from the Activity Management Service, i.e., the scopes of all 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 requested information is transformed to its trustworthy view and delivered to the requester.

B. Implementation Details

The information sharing framework, depicted in Figure 5(a), is designed as distributed service-oriented system, where the single components are implemented as Web services with SOAP and REST interfaces. In this section we highlight implementation decisions, that are further discussed in the evaluation part of this paper.

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 REST-style interface [14]. On the one side, this fact simplifies the integration with the collaboration portal (JSR-168 portlets), 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 I summarizes the available RESTful operations of the Sharing Proxy. 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 community member in the scope of 'scientific dissemination', can be found in *servletURI/disseminationScopeId/anyActivity/anyMember/listInfos&type=paperdraft.xsd*.

Trust Network Provider Interface. Network members retrieve data about connected neighbors in a system-managed trust graph, 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.

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

Table I
SHARING PROXY REST-STYLE INTERFACE.

Operation	servletURI/scopeId/activityId/memberId/listInfos&type=xsd	servletURI/scopeId/activityId/memberId/infoURI
GET	get all information uris (collection overview)	get specific info identified by uri
PUT	–	replace/update existing information (only with same XSD))
POST	create new information (following existing XSD)	–
DELETE	–	delete specific info (if the requester is the registered owner)

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.

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 1 shows an excerpt of the schema of such catalog entries.

```
<?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" maxOccurs="unbounded"/>
   <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>
 ...
</xsd:schema>
```

Listing 1. Catalog entry schema excerpt.

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 1) 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.

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.

According to the excerpt in Listing 2, 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.

```
<?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" maxOccurs="unbounded"/>
   <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" minOccurs="0"/>
   <xsd:element name="recomendation" type="tnValop" minOccurs="0"/>
   <xsd:element name="reputation" type="tnValop" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
 ...
</xsd:schema>
```

Listing 2. Sharing rule schema excerpt.

Sharing View Definitions. The mentioned 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. The output of transformations are HTML fragments that are directly embedded in a dynamic (X)HTML page and rendered in a Portlet of the Collaboration Portal.

Querying Information Collections. In contrast to the extensively discussed case of an already referenced information (identified by a well-known uri), community members will also search the network for larger sets of data (uri/listInfos). For instance, 'who are co-workers of member n_i?',or 'what are the documents of n_i in the dissemination scope?'.

Algorithm 1 Discover information of $type$ in the network

Require: information $type$, requester $requ$
  $sharedInfoXML[] \leftarrow \emptyset$
  for each $e \in$ getInfoCatalogEntries($type$) **do**
    if $\nexists\, jointActivity(requ, e.owner)$ **then**
      continue loop
    end if
    $trustRel \leftarrow$ getTrustRelation($e.owner, requ, e.scope$)
    $rule \leftarrow$ getRule($e.owner, e.type, e.mode, trustRel$)
    $view \leftarrow$ getView($rule$)
    $info \leftarrow$ getInformation($e.uri$)
    $info' \leftarrow$ applyTransformation($info, view$)
    if $info' \neq \emptyset$ **then**
      add($info'$, $sharedInfoXML[]$)
    end if
  end for
  return $sharedInfoXML[]$

Algorithm 1 depicts the order of requests from the Sharing Proxy perspective, when requester $requ$ queries for information of a particular $type$ in the whole network. After retrieving all catalog entries of the $type$ of interest, each entry is processed. The trust relation to the corresponding information owner in the selected scope is evaluated, and configured rules applied (using XSLTs). Finally, all found information is returned in its individually restricted shape.

V. Evaluation and Discussion

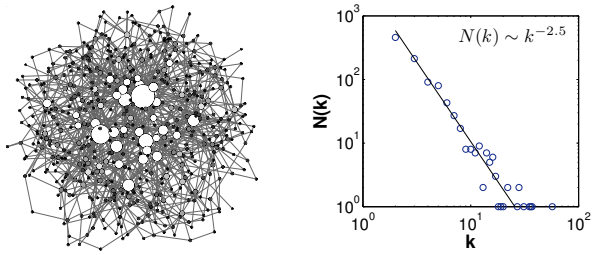
A fundamental aspect of our trust-based adaptation approach is the context-awareness of data and trust. Due to the high complexity of large-scale networks comprising various kinds of interactions, distinct scopes of trust, and large blocks of shared information, we evaluate the feasibility of our framework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

A. Preparation

For conducting our performance studies, we generate an artificial interaction and trust network that we would expect to emerge under realistic conditions. For that purpose we utilize the *preferential attachment model* of Barabasi and Albert to create[2] network structures that are characteristic for *science collaborations* [7]. As shown in Figure 6 for a graph with 500 nodes, the output is a scale-free network with node degrees[3] following a power-law distribution. These structures are the basis for creating realistic interaction logs that are used to conduct trust inference experiments. For a graph $G = (N, E)$, 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 XML artifacts, including some information to be shared, catalog entries, and common sharing rules (accounting for trust and recomendation) and views.

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

[3]the node size is proportional to the degree; white nodes are 'hubs'



(a) scale-free graph structure. (b) power-law node distribution

Figure 6. Generated network using the preferential attachment model.

B. Experiments

For the following experiments, the Sharing Proxy and the backend services are hosted on a server with Intel Xeon 3.2GHz (quad), 10GB RAM, running Tomcat 6 with Axis2 1.4.1 on Ubuntu Linux, and MySQL 5.0 databases. The client simulation runs on a Pentium 4 with 2GB on Windows XP, and is connected with the servers through a local 100MBit Ethernet.

Interaction Logging and Trust Inference. Through utilizing available interaction properties, we calculate three metrics (i) *average response time*, (ii) *success rate* (ratio of success to the sum of success and failure notifications), and (iii) *support reciprocity* (ratio of served to requested support). Individual response times are normalized to $[0, 1]$ with respect to the highest and lowest values in the whole network. Confidence between each pair of connected nodes accounts for all three metrics equally. If the amount of interactions $|I(n_i, n_j)|$ between a pair (n_i, n_j) is below 10, we set the reliability of confidence to $\frac{|I(n_i,n_j)|}{10}$, else we assume a reliability of 1. Trust is calculated (for two independent scopes) by multiplying confidence with its reliability.

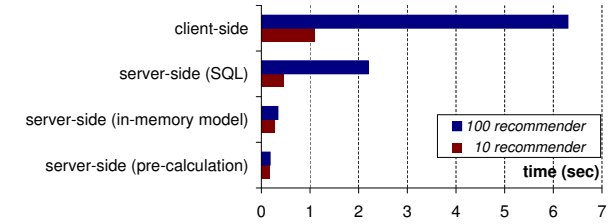
We measure the required time to completely process the interaction logs, including reading logs from the interaction database (SQL), aggregating logs and calculating metrics, normalizing metrics (here only the response time, because the values of other metrics are already in $[0, 1]$), computing trust, and updates in the trust graph (EMA with $\alpha = 0.5$). The results show that especially for medium and large networks only a periodic offline calculation is feasible.

Table II
Trust inference performance results.

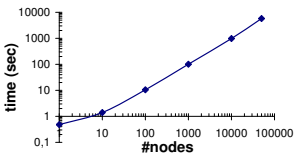
network characteristics	trust computation time
Small-scale: 100 nodes, 191 edges	1 min 44 sec
Medium-scale: 1000 nodes, 1987 edges	17 min 20 sec
Large-scale: 10000 nodes, 19983 edges	173 min 19 sec

Network Management and Trust Provisioning. This second set of experiments, deal with trust provisioning and the calculation of recommendation and reputation on top of a large-scale trust network (10000 nodes). Figure 7(a) depicts the required time in seconds to calculate the recommendation $\tau^s_{rec}(n_i, n_j)$, having 10 and 100 recommender (i.e., intermediate nodes on connecting parallel paths (n_i, n_j)

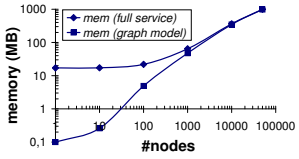
of length 2). Several ways to implement recommendations exist. First, a client may request all recommender nodes and their relations and calculate recommendations on the client-side. However this method is simple to implement on the provider side, it is obviously the slowest one due to large amounts of transferred data. Still retrieving all recommender and relations directly from the backend database, but performing the calculation server-side, dramatically improves the performance. However, this method produces heavy load at the provider and its database and deems not to be scalable. Therefore, we map the network data, i.e., a directed graph model with annotated nodes and edges, in memory and perform operations without the backend database. Since all data is held in memory, the performance of calculating recommendations online is comparable to provisioning of pre-calculated data only. Hence, we design our system with an in-memory graph model, and further measure some aspects of this design decision. Figure 7(b) illustrates required time for mapping the whole graph from the backend database to its in-memory representation. Figure 7(c) shows the memory consumption for instances of different sizes, first for the whole Trust Network Provider Service, and second only for the graph object itself.



(a) different recommendation calculation approaches.



(b) graph mapping time.



(c) memory consumption.

Figure 7. Performance tests for mapping the graph model.

Overall End-To-End Performance and Caching. 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 Service* and *Sharing Rule Service* as the most data-intensive services. Therefore, we studied the overall performance when caching data. In particular, the *Sharing Proxy* implements the strategy of self-pruning cache objects as widely adopted [15].

Figure 8 depicts the required time of the *Sharing Proxy* to process different amounts of concurrent client requests.

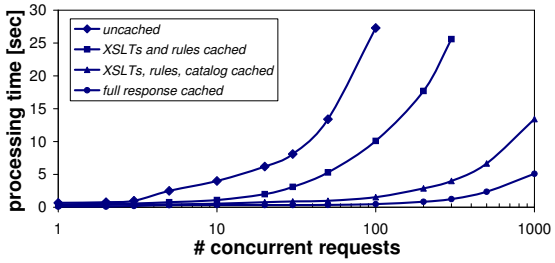


Figure 8. Overall performance of the Sharing Framework.

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 application domain, a trade-off between performance and up-to-dateness of cached data has to be carefully considered.

VI. Related Work

The problem of composition and adaptation is strongly related to organization and control. The autonomic computing paradigm [5] advocates self-* principles to reduce human intervention in configuring systems and services. An autonomic computing environment has the ability to manage itself and dynamically adapt to changes in accordance with objectives and strategies. 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 [5] 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 [16] to guide interactions between actors, the parameters of those strategies, and actions to prevent inefficient use of resources and disruptions.

Recently, trust in social environments and service-oriented systems has become a very important research area. Various surveys of trust in computer science have been performed [1], [2], which outline common concepts of trust, clarify the terminology and show the most popular trust models. Trust management systems for service-oriented environments [17] as well as for mixed systems [4], comprising humans and services, are a focus of current research. They aim at collecting interaction data and user feedback in collaborations of humans and services, and facilitating the emergence of trust among social and collaborative network members. Various works deal with trust models for complex networks, formed by agents [12], [18] as well as services in SOA [17].

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.

Trusted information sharing in the introduced science collaboration network is related to *selective dissemination of information* (SDI) [19], [20]. 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.

VII. Conclusion and Further Work

In this paper we highlighted the application of the widely adopted MAPE approach for adaptations in mixed human-service systems, and discussed the implications of complex interaction networks. Instead of a purely conceptual and algorithmic perspective, we demonstrated the technical grounding, using state-of-the-art Web services technologies. We discussed a framework that supports a concrete use case, i.e., adaptive information sharing in complex research networks, and the underlying concepts, such as information models, trust provisioning, and rule-based adaptation strategies. The evaluation of the running framework discovered important design issues, such as the need for appropriate caching strategies depending on the scale of supported networks. Our approach has important implications on adaptation in complex systems, because it reduces configuration burdens for the user and permits self-regulation of shared information based on collaboration strength.

Future work include the application of our Information Sharing Framework in real end-user scenarios. Therefore, we will integrate the system in the collaboration portal of the COIN[4] project. The focus of COIN is to study new concepts and develop tools for supporting the collaboration and interoperability of networked enterprises. The end-user evaluation in COIN will discover the usability of adaptive information sharing in real business use cases.

Acknowledgment

This work is supported by the European Union through the IP project COIN (FP7-216256).

References

[1] D. Artz and Y. Gil, "A survey of trust in computer science and the semantic web," *Web Semantics*, vol. 5, no. 2, pp. 58–71, 2007.

[2] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[3] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, May 2009.

[4] F. Skopik, D. Schall, and S. Dustdar, "The cycle of trust in mixed service-oriented systems," in *SEAA*, 2009, pp. 72–79.

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

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

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

[8] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *WWW*, 2004, pp. 403–412.

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

[10] F. Skopik, D. Schall, and S. Dustdar, "Trustworthy Interaction Balancing in Mixed Service-oriented Systems," in *ACM Symposium on Applied Computing*, 2010, pp. 801–808.

[11] A. Agrawal et al., "WS-BPEL Extension for People (BPEL4People), Version 1.0." 2007.

[12] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *AAMAS*, vol. 13, no. 2, pp. 119–154, 2006.

[13] Z. Gyngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB*, 2004, pp. 576–587.

[14] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[15] B. D. Goodman, "Accelerate your web services with caching," *IBM Advanced Internet Technology*, December 2002.

[16] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engineering*, 2008.

[17] D. Kovac and D. Trcek, "Qualitative trust modeling in soa," *Journal of Systems Architecture*, vol. 55, no. 4, pp. 255–263, 2009.

[18] J. Caverlee, L. Liu, and S. Webb, "Socialtrust: tamper-resilient trust establishment in online communities," in *JCDL*. ACM, 2008, pp. 104–114.

[19] M. Altinel and M. J. Franklin, "Efficient filtering of xml documents for selective dissemination of information," in *VLDB*, 2000, pp. 53–64.

[20] Y. Diao, S. Rizvi, and M. J. Franklin, "Towards an internet-scale xml dissemination service," in *VLDB*, 2004, pp. 612–623.

[4] http://www.coin-ip.eu