Trustworthy Interaction Balancing in Mixed 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

Web-based collaboration systems typically require dynamic and context-based interactions between people and services. To support such complex interaction scenarios, we introduce a mixed service-oriented system that is composed of both humans *and* software services, collaborating and interacting to perform certain activities. As an example, consider a professional online help and support community spanning interactions between human participants and software-based services. Trust between these members is essential for successful collaborations and has been extensively studied in the context of social and collaborative networks. In this paper, we discuss trust from a collaborative and social point of view instead of a security perspective. Our approach follows an interaction monitoring and an interpretative rule-based trust inference model established on previous behavior.

However, trust relations encourage network members to continue interacting with successful (and thus trusted) collaboration partners, and to avoid, or even refuse, interactions with unknown actors. This behavior has negative side-effects from a global community perspective. Given the help and support environment, a small number of popular network members will become increasingly overloaded with support requests. We solve this load and interaction balancing problem by the means of trustworthy request delegations.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based Services; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Trust, Interactions, Service-oriented Systems

Keywords

Fuzzy Trust Model, Mixed System, Interaction Balancing

1. INTRODUCTION

The support of people's interactions on collaborative and social Web platforms has evolved in a rapid pace over the last few years. Services have become a fundamental pillar of modern Web architectures. Today's support of pervasiveness, context-awareness, and adaptiveness leads to a paradigm shift from traditional closed environments to open, loosely coupled, service-oriented systems. However, as these systems become increasingly complex, they also require new approaches to support interactions.

A mixed service-oriented system comprises human- and software services that can be flexibly and dynamically composed to perform various kinds of activities. Therefore, interactions in such a system do not only span humans, but also software services. Recently, *trust* has been identified as a beneficial concept in large-scale networks [2, 8]. Considering trust relations when selecting people for communication or collaboration, services to be utilized, and resources to be applied leads to more efficient cooperation and compositions of human- and software services [18]. In contrast to many others, we do not discuss trust from a security perspective. In this work we follow another view that is related to how much humans or other systems can rely on services to accomplish their tasks [14].

We adopt common definitions of trust in collaboration environments [5, 12, 18] and define trust as follows:

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

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

- Trust reflects an expectation and, therefore, cannot be expressed objectively. It is influenced by subjective perceptions of the involved actors.

- Trust is context dependent and is basically valid within a particular scope only, such as the type of an activity or the membership in a certain team.

- Trust relies on previous interactions, i.e., from previous behavior a prediction for the future is inferred.

Several works have previously shown [2, 5, 8] that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments. However, trust between human and software services is emerging based on

interactions. Interactions, for example, may be categorized in terms of success (e.g., failed or finished) and importance. Therefore, a key aspect of our approach is the monitoring and analysis of interactions to automatically determine trust in mixed service-oriented systems. We argue that in large-scale SOA-based systems, only automatic trust determination is feasible. In particular, manually assigned ratings are time-intensive and suffer from several drawbacks, such as unfairness, discrimination or low incentives for humans to provide trust ratings. Moreover, in mentioned mixed systems, software services demand for mechanisms to determine trust relations to other services. Much research effort has been spent on defining and formalizing trust models (e.g., see [1, 7, 12]). We present the following novel contributions:

- *Interaction-based Trust Emergence.* We show the establishment of trust based on dynamic interaction patterns [4] in mixed service-oriented environments.

- *Trust Interpretation.* Due to the dynamic aspects in mixed systems, we do not apply a 'hard-wired' and static analytical trust model, but consider the subjective view and nature of trust by applying a rule-based interpretative approach.

- *Implementation and Architecture.* We realize our theoretical concepts with existing Web service standards, accounting for and relying on mechanisms that are typically available in service-oriented systems, such as WSDL service descriptions, and logging of SOAP-based interactions.

The paper is organized as follows. In Section 2 we show the fundamental concepts of trust emergence in mixed service-oriented systems, a motivating example and the challenges of our work. Section 3 describes the approach of trust inference and the utilized trust model. The application of trust relations to enable trustworthy delegations and, therefore, interaction balancing in communities is covered by Section 4. We evaluate our approach with a simulation in Section 5. Section 6 deals with related work and Section 7 concludes the paper.

2. INTERACTIONS AND COMPOSITIONS

We depict a professional virtual community (PVC) environment to familiarize with our concepts, and to demonstrate the emergence of trust. A PVC is a virtual community that consists of professionals and experts who interact and collaborate by the means of information and communication technologies to perform their work. Nowadays, service-oriented technologies are used to realize PVCs. The actors, i.e., the community network members, that are both humans and software services, provide *help and support* on requests of each other. In such a mixed service-oriented environment actors have to register at a central community management service to become part of the network. Humans can register themselves by providing their profiles, including their education, employment status, certified skills and project experience. Services can be registered by their vendors or third party persons that offer information about service features and capabilities.

In the described environment, network members perform activities. Activities are a concept to structure information in ad-hoc collaboration environments, including the goal of
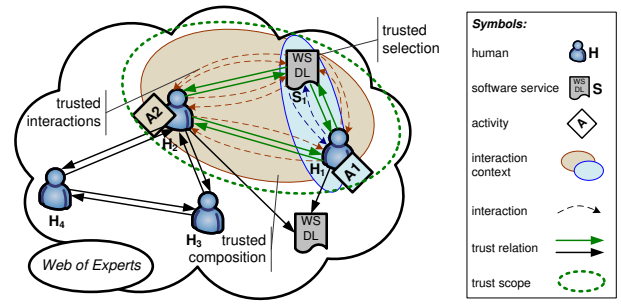

Figure 1: A mixed service-oriented PVC.

the ongoing tasks, involved actors, and utilized resources. They are either assigned from outside the community, e.g. belonging to a higher-level process, or emerge by identifying collaboration opportunities.

In the scenario depicted by Figure 1, the two humans H_1 and H_2 are the owners of activities A_1 and A_2 respectively. We assume activity A_1 is a software implementation activity and A_2 is a software testing activity in some higher-level software development process (not depicted here). The human H_1, requests support from the Web service S_1, that is a software implementation knowledge base, providing code examples and FAQs[1] about software implementation. The dashed arrows represent interactions (requests for support (RFSs)), such as retrieving articles from the knowledge base. Interactions are performed by traditional SOAP calls. Even the capabilities of humans are described by WSDL and communication takes place with SOAP messages (see Human-Provided Services [15]). The interaction context, described by activity A_1 (reflected by the blue-shaded area), holds information about involved actors, goal of the activity, temporal constraints (start, duration, milestones), assigned resources, planned costs, risk with respect to the whole software development process and so on. The detailed description is out of scope of this paper, however, we conclude, that an activity holistically describes the context of an interaction in our environment model [18].

Human H_2, the owner of activity A_2, performs his/her activity (software testing) jointly with the help of H_1 and S_1. For that purpose, s/he interacts with all activity participants, such as requesting help and assigning sub-activities. As defined before, trust emerges from interactions, and is bound to a particular scope. Therefore, we aggregate interactions that occurred in a pre-defined scope, calculate metrics (numeric values describing prior interaction behavior), and interpret them to establish trust. The scope of trust is reflected by the green dashed ellipse in Figure 1. In the given scenario, the scope comprises trust relations between PVC members regarding help and support in 'software development'. So, regardless of whether interactions took place in context of activity A_1 or A_2, interactions of both contexts are aggregated to calculate metrics, because both interaction contexts adhere to the scope of software development. Finally, interaction metrics are interpreted using rules, and the degree of trust between each pair of previously interacting PVC members is determined.

Let us assume we are able to infer meaningful trust between interacting network members (as demonstrated later in this paper). Usually, once a network member becomes

---

[1]frequently asked questions

highly trusted by others (normally leading to globally high reputation), s/he is consulted in future collaborations again and again. Hence, distinguished experts would get overloaded with work and flooded with support requests. We aim at applying a balancing model that relies on the means of delegations. For instance, if network member H_2 is overloaded, s/he may delegate incoming requests (e.g., from H_1) to third, well trusted, network members (e.g., S_1) in the same scope. These third parties may directly respond to the original requester. The delegation model has two important properties:

- *Interaction Balancing.* Interactions are not focused on highly reputed members only, but load is distributed over the whole network.

- *Establishment of new Trust Relations.* New personal trust relations that rely on direct interactions, emerge, leading to future trustworthy compositions.

3. INTERPRETATIVE TRUST INFERENCE

We develop and extend the *VieTE - Vienna Trust Emergence Framework* [18] to research and evaluate novel concepts of trust and reputation in mixed service-oriented system environments. Briefly (see Figure 2), the system captures interactions between network members (bottom layer), calculates metrics of member relations, such as average response time, request success rates, and availability, performs a rule-based interpretation of these metrics, and infers trust between each pair of interacting members (middle layer). Finally a social network, describing collaboration- and trust relationships is provided (top layer). While the depicted architecture follows a centralized approach, the logging facilities are replicated for scalability reasons, and monitoring takes place in a distributed manner. Interactions are purged in predefined time intervals, depending on the required depth of history needed by metric calculation plugins.
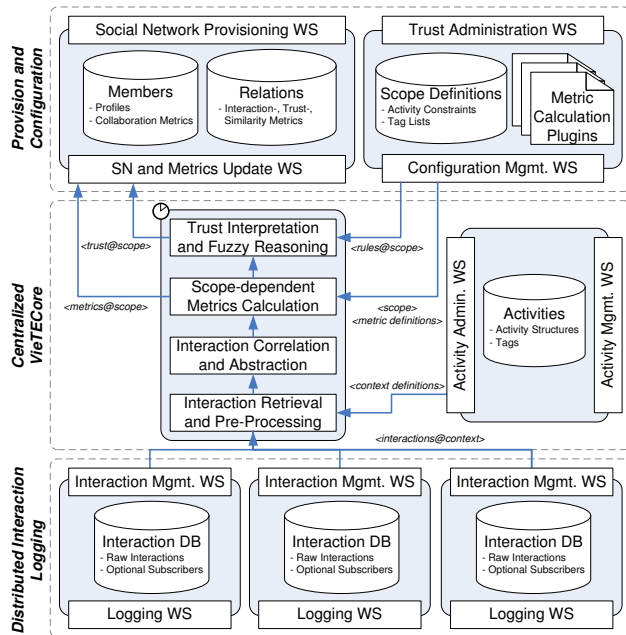

Figure 2: VieTE framework overview.

3.1 Interaction Monitoring

Interactions are captured by interaction sensors, stored and managed by logging services. The requests for support (RFSs) and their responses, exchanged between community members, are modeled as traditional SOAP calls, but with various header extensions, as shown in Listing 1. These header extensions include the context of interactions (i.e., the activity that is performed), delegation restrictions (e.g., the number of hops), identify the sender and receivers with WS-Addressing[2], and hold some meta-information about the RFS itself. For Human-Provided services (HPSs), SOAP messages are mapped to a GUI by the HPS framework [15].

Actors use activities to manage their work as introduced before. Activities are structures to describe work and its goals, as well as participating actors, used resources, and produced project artifacts. A detailed description of this model, used to capture the context of interactions, is provided in [18].

```
<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://hps.infosys.tuwien.ac.at/"
 xmlns:rfs="http://viete.infosys.tuwien.ac.at/Type/RFS">
 <soap:Header>
  <vietypes:timestamp value="2009-03-05T15:13:21"/>
  <vietypes:delegation hops="3" deadline="2009-03-06T12:00:00"/>
  <vietypes:activity url="http://www.coin-ip.eu/Activity#42"/>
  <wsa:MessageID>uuid:722B1240−...</wsa:MessageID>
  <wsa:ReplyTo>http://viete..../Actor#Florian</wsa:ReplyTo>
  <wsa:From>http://viete..../Actor#Florian</wsa:From>
  <wsa:To>http://viete..../Actor#Daniel</wsa:To>
  <wsa:Action>http://viete....ac.at/Type/RFS</wsa:Action>
 </soap:Header>
 <soap:Body>
  <hps:RFS>
   <rfs:requ>Can you create an ant file for projectX?</rfs:requ>
   <rfs:generalterms>programming</rfs:generalterms>
   <rfs:keywords>java, EE, ant, apache axis2</rfs:keywords>
   <rfs:resource url="http://svn.vitalab.tuwien.ac.at/projectX"/>
  </hps:RFS>
 </soap:Body>
</soap:Envelope>
```

Listing 1: Simplified RFS via SOAP example.

3.2 Interaction Metric Calculation

Analyzed *interactions* are RFSs and responses sent by an actor u regarding another one v. The *context* of interactions reflects the situation and reason for their occurrences, and is modeled as activities. When interactions are interpreted, only a minor subset of all describing context elements is relevant within a *trust scope.* In the motivating use case of this paper, such a trust scope may describe the expertise area that is required to process an RFS.

Table 1 shows some example interaction metrics suitable for trust interpretation that can be calculated from logged SOAP calls. Note, as described before, these metrics are determined for particular scopes. The availability of a service, either provided by humans or implemented in Software, can be high in one scope, but much lower in another one. Furthermore, these metrics are calculated for each directed relation between pairs of network members. An actor u might serve v reliably, but not a third party w.

---

[2]http://www.w3.org/Submission/ws-addressing/

Table 1: Some metrics utilized for trust inference.

metric name	range	unit	description
availability	0-100	%	ratio replied/unreplied RFSs
response time	0-96	hours	average response time in hours
success rate	0-100	%	amount of successful RFSs
experience	0-∞	1	number of RFSs served
man. reward	0-5	1	manually assigned scores
costs	0-5	$	price for serving RFSs

For the sake of brevity, in the following examples and evaluation we account only for the *average response time* t_r (Eq. 1) of a service and its *success rate sr* (Eq. 2). These are typical metrics for an *emergency help and support environment*, where fast and reliable support is absolutely required, but costs can be neglected. We assume, similar complexity of requests for support (RFS) in a scope s, thus different RFSs require comparable efforts from services (similar to a traditional Internet forum).

The response time is calculated as the duration between sending (or delegating) a request (t_{send}) to a service and receiving the corresponding response ($t_{receive}$), averaged over all served RFSs. Unique IDs of calls (see SOAP header in Listing 1) enable sophisticated message correlation to identify corresponding messages.

$$t_r^s = \frac{\sum_{rfs \in RFS} (t_{receive}(rfs) - t_{send}(rfs))}{|RFS|} \quad (1)$$

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

$$sr^s = \frac{num(sRFS)}{num(sRFS) + num(fRFS)} \quad (2)$$

3.3 Interpretation and Trust Inference

On top of the interaction metrics M of u towards v in scope s (here: $M = \{t_r^s, sr^s\}$), personal trust $\tau^s(u,v) \in [0,1]$ is inferred. Trust, describing the relationship from u to v, represents recent evidence that an actor behaves dependably, securely and reliably. The function Ψ^s (Eq. 3) evaluates metrics M with a rule set R to interpret trust τ in scope s.

$$\tau^s(u,v) = \Psi^s(u, M(u,v), R, s) \quad (3)$$

Instead of usual business rules, we utilize a fuzzy set theory approach that enables an intuitive way to combine and interpret various metrics as trust from a collaborative and social point of view. Fuzzy set theory, developed by Zadeh [22], and fuzzy logic emerged in the domain of control engineering, but are nowadays increasingly used in computer science to enable lightweight reasoning on a set of imperfect data or knowledge. The concept of fuzziness has been used earlier in trust models [6, 13, 17], however, to our best knowledge not to enable an interpretation of trust from larger and diverse sets of metrics, calculated upon observed interactions. Due to space limitations we do not outline fuzzy set theory here, but refer to further literature, for instance [23].

A linguistic variable comprises a set of membership functions that describe the 'grade of membership' of an element to a fuzzy set. Figure 4 shows examples for the linguistic variables *response time* t_r and *success rate sr*. The corresponding membership functions describe, for instance, what means low, medium, and high *response time*, and the transitions between these sets. An example, showing two rules

that fire, and how concrete input values are fuzzified, interpreted and defuzzyfied again, is depicted.

```
if tr is low    and sr is high   then trust is full
if tr is low    and sr is medium then trust is high
if tr is low    and sr is low    then trust is low
if tr is medium and sr is high   then trust is high
if tr is medium and sr is medium then trust is medium
if tr is medium and sr is low    then trust is low
if tr is high   and sr is high   then trust is medium
if tr is high   and sr is medium then trust is low
if tr is high   and sr is low    then trust is low
```

**Listing 2: Rules for inferring trust upon t_r and sr.**

Example: Given the linguistic variables *response time* t_r, *success rate sr*, and *trust*, with the membership functions as defined in Figure 4, we provide the rulebase in Listing 2 to the fuzzy engine. Figure 3 visualizes trust inference results for different pairs of t_r and sr inputs.
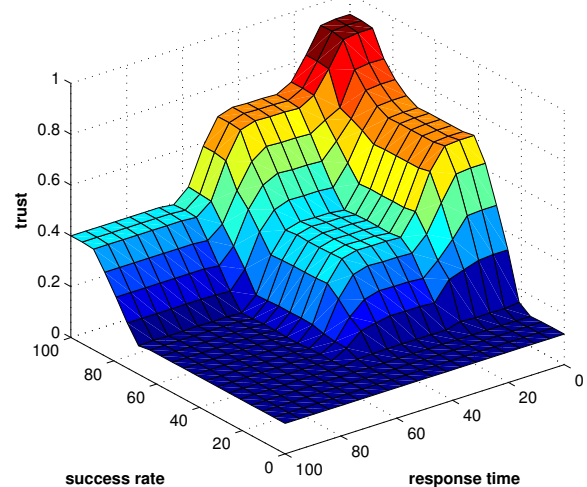
Figure 3: Result space for the given rule set.

Personal trust $\tau^s(u,v)$ from u in v is updated periodically in consecutive time intervals (e.g., on a daily basis). We apply a sliding window approach and process all logged interactions within a pre-configured time frame (e.g., the last week or month). The size of the window depends on the calculated interaction metrics. For instance, success rates or collected experiences are inferred from interactions within the last six month, while response times only depend on interactions in the last week. Therefore, the depth of the required interaction history relies on the utilized metrics of the environment. We demonstrate an application of this approach in the evaluation in Section 5.

3.4 Collaboration Network Provisioning

Finally, the social network, comprising actors connected by trust relations, is provided by VieTE through a SOAP interface (see top of Figure 2). A trust relation is always asymmetric, i.e., a directed edge from one member node to another one in a graph model $G = (N, E)$. We call the trusting actor the *trustor* u (the source of an edge), and the trusted entitiy the *trustee* v (the sink of an edge). VieTE's provisioning interface, described by WSDL, supports convinient network retrieval operations, such as getting the trustors and trustees of a node in a specified scope.
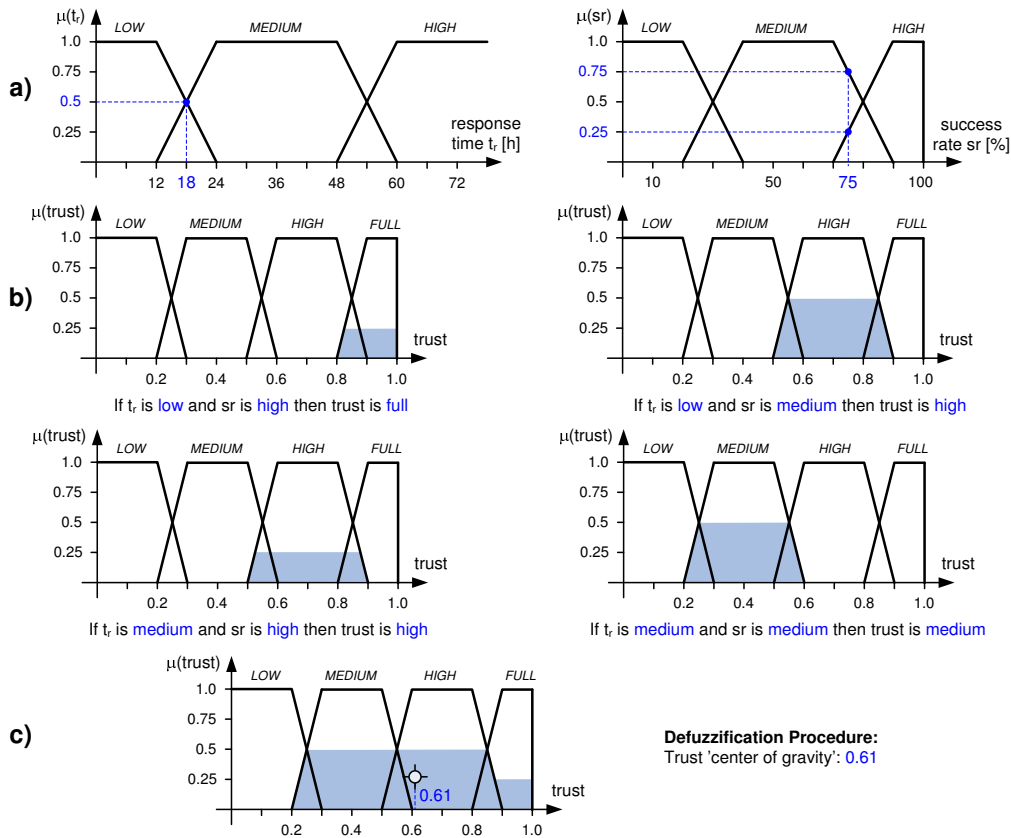
Figure 4: An example showing fuzzy trust inference. Applied interaction metrics are response time $t_r = 18h$ and success rate $sr = 75\%$. (a) definition of membership functions and fuzzified interaction metrics; (b) four applied fuzzy rules following the max-min inference; (c) defuzzification by determining the center of gravity.

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

4. DELEGATIONS AND BALANCING

A common problem of trust and reputation mechanisms in online communities is that there emerge only a minority of highly trusted actors, while the majority remains in the background. Therefore, network members tend to consult and interact with the same (already trusted) services again and again, leading to work overloads of these service providers, and hindering the emergence of new trust relations. We utilize the means of delegations to compensate this load and interaction balancing problem that is often neglected, but of paramount importance in collaborative environments. In the motivating PVC scenario of this paper actors send and process requests for support (RFS). Once an actor gets overloaded s/he should be able to delegate requests to other actors (with potentially free resources). If the receivers of such delegations behave trustworthy, i.e., respond fast and reliably, the original requesters will establish trust to them. Figure 5 visualizes this model. In case of a

successful delegation, u sends an RFS to v who delegates to w, and w responds directly to u. This interaction will positively impact the metrics that describe the relation from u to w, and finally $\tau(u,w)$ increases. The relation $\tau(u,v)$ is neither rewarded nor punished, because on the one side v did not serve u's RFS, but on the other side, v was able to successfully delegate, and thus did not harm u. The relation $\tau(v,w)$ is also not influenced, since v is not the original requester. If a delegation fails (Figure 5(b)), i.e., an RFS is not responded, metrics that describe both $\tau(u,v)$ and $\tau(v,w)$ are negatively influenced (for instance the success rate is decreased), because of v's and w's unreliable behavior. But in that case, we assume that $\tau(u,w)$ remains unchanged. Although w has not served u's request, we do not know the reasons for that behavior. For instance, a denial of service attack could maliciously harm w's reputation (the sum of trust relations), if s/he is flooded with delegated RFSs.

(a) Successful delegation   (b) Failed delegation
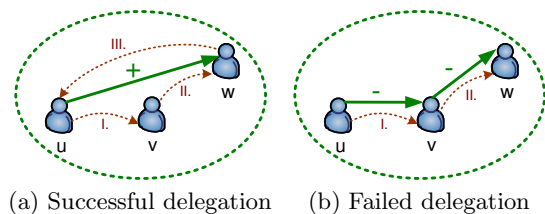
Figure 5: Delegations and their impact on trust.

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

One of the major challenges to enable sophisticated balancing is to determine the receivers of delegations in the whole network. Usually, the selection will rely on trust, because, as shown in Figure 5(b), it is in v's interest to delegate successfully and not to get punished. A fundamental selection strategy randomly picks an actor from a pool of service providers that are personally trusted above a predefined limit. Based on each individual's interaction history, every network member has his/her own pool of trusted actors. More advanced selection models are out of scope of this paper, and are subject to further research.

5. EVALUATION OF THE PVC SCENARIO

We evaluate the VieTE framework that implements our approach of fuzzy trust inference and balancing, by simulating typical scenarios in the described PVC environment. For that purpose, we utilize the popular Repast Simphony[3] toolkit, a software bundle that enables round-based agent simulation. In contrast to researchers in the agent domain, we do not simulate our concepts by implementing different actor types and their behavior only, but we use a network of actors to provide stimuli for the actual VieTE framework. Therefore, we are not only able to evaluate our new approach of fuzzy trust inference, but also the technical grounding based on Web service standards. Figure 6 depicts that VieTE is used exactly in the same manner in our simulation (highlighted left side), as it would be used by a real mixed systems PVC (right side).
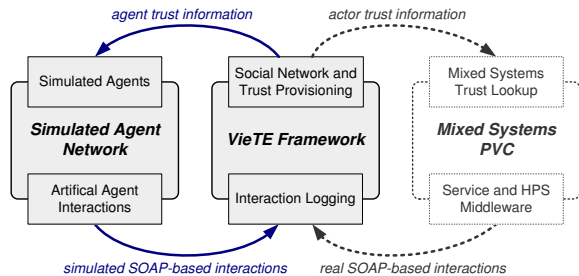
Figure 6: Simulation setup (left side) compared to the intended usage (right side) of VieTE.

In particular, we let the simulated network members interact (sending, responding, and delegating RFSs), and provide these interactions to the logging facilities of VieTE. The framework infers trust by calculating the described metrics t_r and sr, and using the rule set of Listing 2 for behavioral interpretation. Finally, emerging trust relations between the simulated actors influence the selection of receivers of RFSs. Hence, VieTE and the simulated actor network relies on each other, and are used in a cyclic approach; exactly the same way VieTE would be used by real PVCs. To facilitate simulations, all interactions take place in the same scope.

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

5.1 Simulation Setup

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

We set up a network comprising 15 actors, where only one is highly reputed and fully trusted by all others as depicted in Figure 7(a). This is the typical starting point of a newly created community, where one actor invites others to join.

VieTE Setup. After each simulation step (round) seven randomly picked actors send one RFS to its most trusted actor (in the beginning this will only be the highly reputed one who starts to delegate). Each actor's input queue has exactly 5 slots to buffer incoming RFSs. A request is always accepted and takes exactly one round to be served. An actor processes an RFS itself if it has a free slot in its input queue, otherwise incoming RFSs are delegated to randomly picked trusted ($\tau > 0.8$) neighbors in the network. Note, one actor does not delegate more than one RFS per round to the same neighbor, however, an actor may receive more than one RFS from different neighbors in the same round. Delegations require one additional simulation round. There is an upper limit of 15 rounds for an RFS to be served (deadline); otherwise it is considered failed. A request can be delegated only three times (but not back to the original requester) (*hops*) to avoid circulating RFSs. Because the simulation utilizes only two fully automatically determined metrics (t_r and sr), and no manual rewarding of responses, we assume an RFS is successfully served if a response arrives within 15 rounds (no fake or low quality responses). After each round, VieTE determines t_r based on interactions in the last 25 rounds, and sr upon interactions in the last 50 rounds (sliding window approach), and purges older logs.

5.2 Simulation Results

Interaction Balancing. We perform 250 simulation rounds of the described scenario with the aforementioned properties, and study the network structure in certain points of the simulation. The depicted networks in Figure 7 show actors with different behavior and the temporal evolvement of trust relations between them. The size of the graph's nodes depend on the amount of trust established by network neighbors. Beginning with a star structure (Figure 7(a)), the network structure in Figure 7(b) emerges after 100 rounds, and Figure 7(c) after 250 rounds respectively. Note, since the behavior of the nodes is not deterministic (i.e., RFSs are sent to random neighbors that are trusted with $\tau > 0.8$ (lower bound of *full trust*; see Figure 4)), the simulation output looks differently for each simulation run, however, the overall properties of the network are similar (number and strength of emerged trust relations).
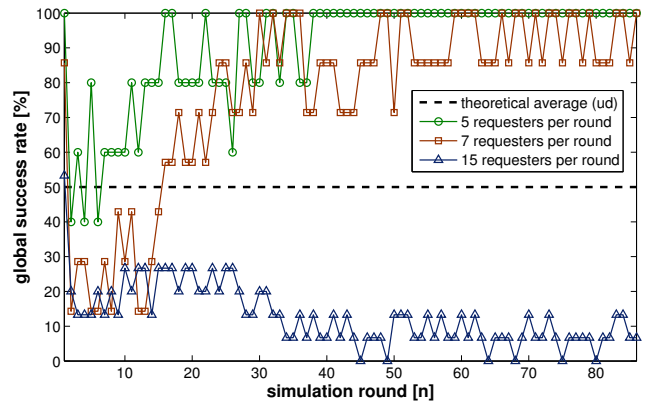
In the beginning, all RFSs are sent to *actor 0*, who delegates to randomly picked trusted actors. If they respond reliably, then requesters establish trust in that third parties. Otherwise they lose trust in *actor 0* (because of unsuccess-

(a) initial n=0 (b) intermediate n=100

(c) balanced n=250 (d) balanced (reduced)
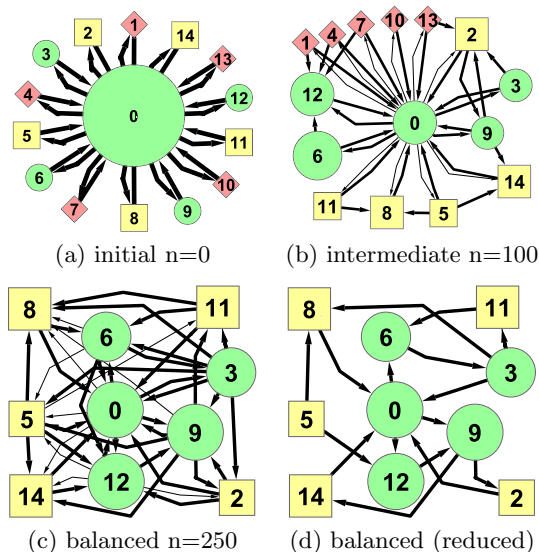
Figure 7: Network structure after simulation round n={0, 100, 250}. Elliptic nodes are fair players, rectangular shapes represent erratic actors, diamond shaped nodes reflect nodes with malicious behavior.

Figure 8: Global RFS response success rate of the simulated actor network.

ful delegations). Therefore, actors with even-numbered IDs lose trust in *actor 0* faster than odd-numbered actors, because if *actor 0* delegates requests to erratic actors, they are not replied. As an additional feature in round 100, actors that are not trusted with $\tau > 0.2$ by at least on other network member, are removed from the network, similar to Web communities where *leechers* (actors that do not contribute to the network) are banned. Therefore, actors with malicious behavior disappear, while actors with erratic behavior still remain in the network. Figure 7(d) shows a reduced view of the balanced network after 250 rounds. Only trust relations with $\tau > 0.8$ are visualized. As expected most nodes have strong trust relations in at least one fair player (actors who reliably respond and delegate RFSs). However, remember that erratic actors reliably serve only requests coming from actors with odd-numbered IDs. Therefore, *actor 3* and *actor 9* also establish full trust in actors from this class. Note, if *actor 3* and *actor 9* would have re-delegated much RFSs coming from even-numbered actors to erratic actors, than those RFSs would have failed and only low trust would have emerged. However, due to the comparatively low load of the network (less than half of the actors receive RFSs per round (until $n = 100$)), only a low amount of re-delegations occur (approx. 8 percent).

Global Success Rate. We run the simulation with different interaction rates of actors. In particular, we let 5, 7 (as in the experiment described before), and 15 actors send RFSs to others, and calculate the global success rate, i.e., the amount of successfully answered RFSs from a global point of view. If requests would not be sent to trusted actors that proved their reliable behavior before, but uniformly distributed over available actor classes (5 fair, 5 erratic, 5 malicious) – according to a primitive interaction balancing approach – than 50 percent of RFSs would be served successfully. This theoretical limit (without delegations) is represented as a reference in Figure 8 by the dashed line. We study the performance of our trustworthy interaction bal-

ancing approach compared to this primitive method.

In Figure 8 some results are visualized, and the following issues are worth mentioning. The deadline for an RFS to be served is 15 rounds. So, the success rate of round i is available in round $i + 15$. We simulate the actor behavior until round 100 (where malicious actors are banned). Therefore, the temporal evolution of the global success rate is depicted until round 85. In the beginning, i.e., the very first round, the success rate is very high, because all RFSs are sent to *actor 0* (a fair player), who performs/buffers the first five RFSs itself (and then begins to delegate). Hence, the success rate is high for the first round, but collapses in the second round, where virtually only delegations occur. The rate slowly recovers as new trust relations emerge, and members get connected to new potential actors to send RFSs to.

In the simulation, where each of five randomly picked actors send one request to another one, the success rate stabilizes at a high level, after new trust relations emerged, and the network has been balanced. This process can be studied for seven actors again. However, since it comes to re-delegations, some RFSs frequently fail being processed (due to the impact of erratic actors), and therefore, the success rate oscillates at a high level. The case of 15 interacting network members per round shows, that actors are mainly busy with re-delegations and the large part of RFSs miss their deadlines. This results in a much lower success rate than the theoretical average of 50%.

Finally, note that our interaction balancing model of trustworthy delegations performs best, if the overall amount of reliable actors is high, and the load is low to medium.

6. RELATED WORK

The role of interactions in large-scale collaborative networks has been studied and discussed by [4]. Interaction concepts, such as specific interaction and delegation patterns, are applied in mixed service-oriented systems. Mixed Systems, consisting of humans and software services, are realized with SOA concepts, regarding service discovery, service descriptions (WSDL), late dynamic binding, and SOAP-based interactions. In such networks, humans participate and provide services in a uniform way by utilizing the Human-provided Services framework [15]. The importance of interactions and their impact on trust in large-scale mixed systems has been described earlier in [18].

Trust as a computational concept has been extensively studied first by Stephen Marsh [10]. Since that, further widely recognized works in computational models have been performed, such as [5, 12]. While some works, including [3], underline social aspects and subjective components of trust, others research the detection of attacks and malicious behavior using appropriate trust models [19]. Trust models highlight concepts from either an abstract perspective, i.e., not bound to specific scenarios or environments, or show their application in certain domains, such as behavior in agent networks [21] or service-oriented environments [11]. In particular, in our paper we construct a trust model that is closely connected to and applied in the SOA domain. This trust model relies on mechanisms of SOA, such as SOAP-based interaction monitoring, or Web services based trust provisioning. Other works in that domain, such as [9, 16, 20] disregard the human factor in large-scale networks.

Fuzzy set theory has been applied in trust models before [6, 13, 17], however, to our best knowledge, not to interpret diverse sets of interaction metrics. Utilizing interaction metrics, in particular calculated between pairs of network members, enables us to incorporate a personalized and social perspective. For instance, an actor's behavior may vary toward different network members. This aspect is usually out of scope in Web Services trust models, that are often closely connected to traditional QoS approaches. Moreover, most trust models in the SOA domain utilize trust for service selection only (for instance see [11]), and neglect the collaborative aspects and the human factor.

7. CONCLUSION AND FURTHER WORK

In this paper we discussed the VieTE framework, in particular its flexible and interpretative trust model for mixed service-oriented systems. We identified the interaction balancing problem in ad-hoc collaboration scenarios, and the demand to support the establishment of new trust relations between unknown actors. Unlike other approaches that predict not existing relations by the means of trust propagation (transitive relations), recommendation, and reputation, we facilitate the emergence of new personal trust relations. This is achieved by connecting concepts of delegations to our trust model.

Currently we are working on a SOA testbed environment that will enable experiments in a real mixed service-oriented system. Our trust model will be utilized as a foundational pillar for collaboration support in the EU FP7 COIN[4] project.

Acknowledgments

8. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *HICSS*, 2000.

[2] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semantics*, 5(2):58–71, 2007.

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

[4] S. Dustdar and T. Hoffmann. Interaction pattern detection in process oriented information systems. *DKE*, 62(1):138–155, jul 2007.

[5] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[6] N. Griffiths. A fuzzy approach to reasoning with trust, distrust and insufficient trust. In *CIA*, volume 4149, pages 360–374, 2006.

[7] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *AAMAS*, 13(2):119–154, 2006.

[8] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[9] D. Kovac and D. Trcek. Qualitative trust modeling in soa. *Journal of Systems Architecture*, 55(4):255–263, 2009.

[10] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, April 1994.

[11] E. M. Maximilien and M. P. Singh. Toward autonomic web services trust and selection. In *ICSOC*, pages 212–221, 2004.

[12] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *HICSS*, page 188, 2002.

[13] S. Rajbhandari, O. F. Rana, and I. Wootten. A fuzzy model for calculating workflow trust using provenance data. In *ACM Mardi Gras Conference*, page 10, 2008.

[14] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2), May 2009.

[15] D. Schall, H.-L. Truong, and S. Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, 2008.

[16] P. D. Sharon Paradesi and S. Swaika. Integrating behavioral trust in web service compositions. In *ICWS*, 2009.

[17] W. Sherchan, S. W. Loke, and S. Krishnaswamy. A fuzzy model for reasoning about reputation in web services. In *SAC*, pages 1886–1892, 2006.

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

[19] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *WWW*, pages 422–431, 2005.

[20] M. G. Uddin, M. Zulkernine, and S. I. Ahamed. Collaboration through computation. *SOCA*, 3(1):47–63, 2009.

[21] Y. Wang and M. P. Singh. Trust representation and aggregation in a distributed agent system. In *AAAI*, 2006.

[22] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[23] H.-J. Zimmermann. *Fuzzy Set Theory— and Its Applications*. Kluwer Academic Publishers, third edition, 1996.

---

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