The Cycle of Trust 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*—Many collaboration platforms are realized as service-oriented systems enabling flexible compositions of services and support of interactions. Interactions between entities in such systems do not only span software services, but also human actors. A mixed service-oriented system is therefore composed of human and software services. In open environments, interactions between people and services are highly dynamic and often influenced by the role and reputation of collaboration partners. In this paper we present an architecture for the management of trust in such mixed systems environments. In contrast to traditional solutions that typically focus on the matching of actors' skills and competencies with collaboration requirements only, we propose a trust-based 'feedback loop' enabling the inference and consideration of trust relationships based on observed interactions. This cycle, spanning interaction monitoring, trust analysis, trust-enabled collaboration planning, and trust-supported execution of activities and tasks, permits dynamic and trust-aware collaborations in service-oriented environments.**

***Keywords*-trust cycle; trust management; mixed system; human and service interactions**

I. Introduction

Service-oriented architectures (SOA) were understood as environments where software services are registered, discovered, and invoked. This limited perspective on SOA has changed over the past years because it has been realized by researchers and industry players that humans must be part of such systems. Not only interactions between software services and the ability to compose complex flows are important, but rather the interplay between compositions of human and software services. We define such environments as *mixed systems* comprising software services and Human-Provided Services (HPS, see [1]).

A key requirement for the success of mixed systems is the ability to support interactions in a flexible yet reusable manner. In this work, we focus on an activity-centric approach to support and manage interactions between various actors including people and software services. In contrast to many existing approaches based on workflow systems, activities are not predefined, for example by an administrator that is responsible for the design of a workflow. Activities can be modeled as templates before collaborations and interactions start; however, activities are flexible and can be reorganized by adding, removing, or delegating activities at run-time. Hence, activities are essential to allow for dynamics in interactions and to structure collaborations establishing the link between artifacts, services, and people. Existing work [2], [3], [4] has studied activities as a concept to cope with the challenges of dynamic collaboration environments.

Activity-centric collaboration is useful in many different application domains ranging from business-centric collaboration to large-scale social networking platforms. In enterprises, an activity management platform can be used to deal with the increasing complexity in coordination and communication. For example, an activity workspace allows people to manage their interactions (status, task progress, or artifact changes) as well as various services that can be used in the context of activities (e.g., communication services). On the other hand, people using social networking platforms might use activities in a much more informal way. Activities in a social context may depict 'simple' structures and flows, which are in a manner similar to user defined data or mashups of services. Such activity-based templates can be shared with other users to support collaborations.

We strongly believe that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments. However, trust between human and software services is emerging based on interactions. Interactions, for example, may be categorized in terms of success (e.g., failed or finished). Therefore, an important aspect of our approach is the monitoring and analysis of interactions to automatically determine trust in mixed systems.

In this paper we present the following key contributions:

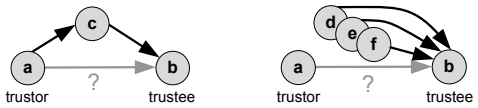
- We establish a detailed understanding of trust in mixed (service-oriented) systems.
- We present our approach for the context-aware analysis of interactions to establish trust. The Vienna Trust Emergence Framework (VieTE) [5] comprises the most important features for the management and calculation (mining) of trust.
- A graph-based algorithm to perform trust mining over heteregenous sources of captured interaction data.

In Section II we start with the definition of trust models for context-based interactions in mixed systems. Our monitoring and analysis approach is inspired by a feedback loop (Section III) that is typically applied to self-organizing and managing systems. An architecture and implementation overview of our VieTE framework is given in Section IV, and an end-user perspective discussed in Section V. We outline related work and approaches in Section VI, and conclude and present future work in Section VII.

II. Trust Modeling in Collaborations

A. Entity Roles and Trust Relations

In trust research the roles of entities in a directed trust relation are often defined as **trustor**, which is the trusting entity, and **trustee** which is the trusted entity [6].



(a) Recommendation. (b) Reputation.

Figure 1. Trust relations.

We distinguish between different kinds of trust relationships in a network of collaborating entities.

Direct Trust Relations. These relations base on first-hand experiences and are inferred from the success and outcome of previous interactions between the trustor and the trustee.

Recommendations. These relations, based on second-hand experiences, are inferred from the success and outcome of previous interactions between a well trusted entity and the trustee. This case is depicted in Figure 1(a), where the relation from a to b is derived by considering the relations from a to c and from c to b, ultimately c recommends b to a.

Reputation. Reputation is a concept where trust of the trustor to the trustee is completely inferred from third party relationships as depicted in Figure 1(b). By considering trust of d, e, and f in b, a may derive a notion of trust in b. Even though reputation is the most unreliable way to determine trust (compared to first-hand experiences or recommendations), it is nevertheless a useful concept, especially the more third party relations to the trustee are considered.

B. Context of Trust

Trust is context dependent, which means trust relations cannot be determined generally, but with respect to a particular situation [7], [6], [8]. Generally each situation, which is determined and described by context data, is unique and consists of multi-dimensional properties.

We developed a system for trust inference in collaborative environments; thus we are able to reduce the complexity of context elements. We define that collaboration situations are basically reflected by describing the actually performed activity, including the goal and nature of work. In a simplified case, when considering only the type of activities performed, a human trustor might trust a trustee to *organize a meeting* (context 1), but not to *develop a software module* (context 2). In this case contexts are obviously different and not closely related to each other. However, there are cases where contexts may be similar. For instance, the trustor might have established trust to the trustee regarding *software implementation*. This means the trustor can trust the trustee to perform assigned activities in the area of *software implementation* reliably in the given time with the required quality. If the trustor wants to know how much s/he can trust the trustee with respect to *software testing* activities, trust can be inferred from the relation regarding *software implementation*, because both *implementation* and *testing* are part of software development, and thus activities in both fields will typically have similar requirements. Hence, the concept of trust context allows (i) distinguishing trust relations with respect to different contexts and thus, expressing trust relations more precisely and reliably, and (ii) deriving relations for new situations based on relations in other, but similar situations. We call the second case *trust propagation over contexts*. This concept permits considering trust relations established in different contexts by taking the similarities between contexts into account.

C. The Diversity of Trust

Trust is a diverse concept and relies on various impacting factors. Figure 2 depicts the most important influences on the trust relationship from a *trustor* to a *trustee* with respect to *context 1*.

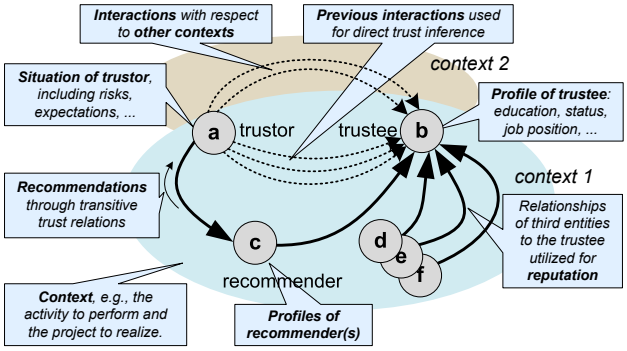


Figure 2. Factors influencing trust.

The *profile of the trustee*, such as the educational status and job position, is utilized, e.g., when it comes to collaboration partner selection or activity assignment. The profile describes if the trustee owns the formal competencies to be trusted to perform a given activity reliably. The current *situation of the trustor*, such as his risks and expectations with respect to a particular situation is a further influencing factor. Decisions about whom to trust always depend on the risks the trustor is facing. If the trustor wants to assign the trustee a particular activity, but the negative consequences for the trustor are high in case the activity is not performed well, then the trustor will not trust the trustee carelessly. The *context*, describing the activity to perform or the overall project to realize, is used to determine which *previous interactions* have to be aggregated to determine the direct trust relation. *Interactions with respect to other contexts*, can be utilized for trust inference as well (however with reduced relevance), especially if previous situations and current situation share similar contextual properties. Furthermore, direct *recommendations* from well-known and trusted recommenders are taken into account. This concept utilizes the transitivity of trust and is known as trust propagation. Experiments have shown that this concept works well in several cases

[9], [10]. The *profiles of recommenders*, may represent a valuable source of data, to decide if and how much someone's recommendation can be trusted. The *reputation of the trustee*, is defined as the aggregated community trust and determined by the sum of all trust relations from different entities in the trustee. Considering reputation only is the most unreliable kind of determining trust, because it does not take personal preferences into account. However, even if the trustor has not established relations to the recommending entities, a reasonable notion of trust can be inferred when relying on large sets of opinions.

III. From Interactions to the Cycle of Trust

Before we show our new approach for dynamic trust inference, we introduce the most important applied concepts.

A. Supported Interactions

We define a *mixed systems environment* which consists of actors of different types, including humans, services, and also humans offering their capabilities as services (Human-Provided Services [1]), interacting with each other to perform a given set of activities contributing towards a common goal. In this environment, we distinguish between the following interactions: (a) *Human-service interactions:* services can be used to realize collaboration functions in a flexible manner. Humans are able to use services (the representation frontend) to perform collaborations. Typical human-service interactions include map services, document sharing service, etc. (b) *Human interactions as part of software service compositions:* many service interaction scenarios demand for human interactions. A popular example is BPEL4People [11]. (c) *Human interactions using Human-provided Services:* Our previous work included the support of service-human interactions (e.g., see [1]), allowing humans to express and offer their capabilities as services. (d) *Interactions between software services:* such interaction scenarios are found in compositions of software services. For example, output of service A is used as input by service B. (e) *Service initiated interactions towards humans:* such scenarios include notifications or news feeds, which are pushed toward humans.

Besides capturing the type, each interaction is described by particular properties, such as its context, success, exchanged data, and time constraints.

B. Interaction Context

Our system is based on the notion of activities. Activities represent work done and actions performed by actors of different type. The activity model, as shown in Figure 3, allows collaborations to be structured in a flexible way.

The model supports dynamic collaborations in mixed systems because activities can be created at runtime. This is needed to support more flexibility and agility in interactions. An activity may comprise a set of sub-activities which can be assigned to different actors. Activity templates describe the 'how-to' execute a particular activity and what services to be used to work on that activity, and

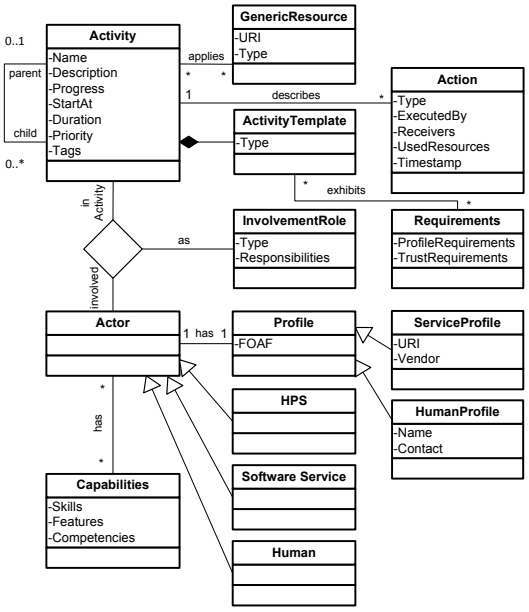


Figure 3. Activity model.

are associated with requirements that are used to control the assignment of actors. Activities can be organized in sets to assemble more complex activities out of the simple ones. The execution of activities and services has to be based on the context of the environment. In activity-centric collaboration, context is any information that lets the actor understand and estimate the current situation, work environment, and the information that is relevant given the actor's current activity. For example, an actor working on an activity needs to know details such as the description, progress and start date of the activity, associated artifacts and their status; information regarding tools, services, and available devices; and context details regarding co-actors (presence status or location information), and finally discussions that are relevant for an activity. However, services are not only used as resources to accomplish activities; instead services play an active role in compositions. For example, services can trigger notifications or reminders based on context information.

During the execution activities can also be delegated to change the ownership and responsibility for a specific activity. Actors perform different types of actions, such as coordinating work, discussing with other actors, or executing some kinds of tasks. The combination of executed actions and performed activities appropriately reflects the situation of interacting actors for trust inference.

C. The Cycle of Trust

Our system design follows the MAPE approach [12], as depicted in Figure 4. MAPE, incorporating basic concepts from the control engineering domain, describes a cycle consisting of four phases, which are *monitor*, *analyze*, *plan* and *execute*. Periodically running through these four phases establishes a kind of environmental feedback control, and therefore, allows to adapt to varying circumstances. Applied in our environment, we are able to infer trust dynamically during ongoing collaborations.

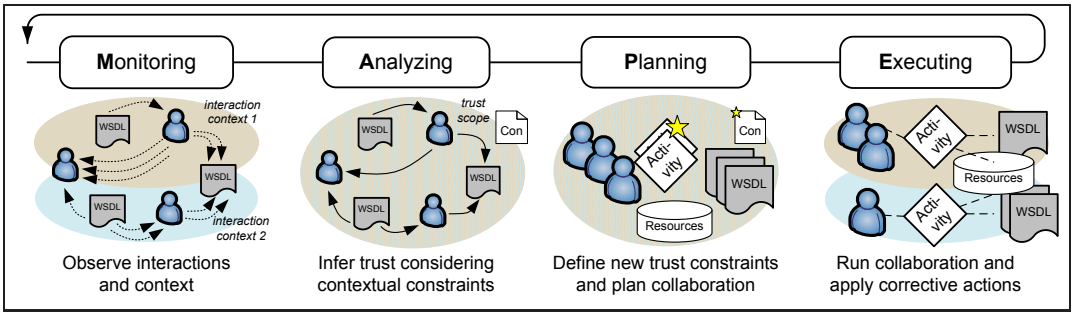


Figure 4. The MAPE cycle applied for dynamic trust inference.

In the **Monitoring Phase** our system observes ongoing collaborations, particularly the creation of new activities, actions executed therein, and interactions taking place, according to the previous descriptions, between humans, services, and HPSs. These interactions, including their types and contexts, are captured and modeled as an interaction network utilized in further trust analyzes. In the **Analyzing Phase** the created interaction network is used to infer trust relationships. For this purpose, the relevance of each interaction is graded automatically considering configurable trust constraints (Con). These constraints define the *scope of trust* and depend on the purpose of trust inference, e.g., constraints differ when calculating trust in an actor to fulfill a particular activity type, or trust in an actor to hold a particular involvement role. Direct trust relationships are calculated by evaluating interactions and their relevance in the defined scope. Based on these direct trust relationships, the concepts of recommendation and reputation are applied to establish relationships relying on second-hand experience and collective opinions. The following **Planning Phase** covers the set up of collaboration scenarios taking the inferred trust relations into account. Furthermore, trust constraints for the next run of the cycle are configured with respect to planned activity requirements. This means it is set up which contextual properties have to be taken into account when calculating trust. This depends on the planning use case, such as the composition of actors, the assignment of roles and responsibilities, the assignment and scheduling of activities, and sharing of artifacts. The **Execution Phase** provides support to enhance the execution of activities, including observing activity deadlines, solving activity scheduling conflicts, checking the availability of actors, and compensation of resource limitations. Furthermore, in parallel the collaboration behavior of actors is monitored, including their actions and interactions. This closes the cycle of trust.

IV. Architecture and Implementation

We developed the VieTE architecture depicted in Figure 5, that has been developed utilizing the MAPE approach. Users, logged in to the management portal, can register new humans and services by entering their profile data. Furthermore, they manage own activities and specify trust constraint sets. On system level, interactions are monitored, captured by software sensors, and analyzed using the configured trust constraints. Recent analysis results lead to the formation of new trust relations, and impact existing ones. This knowledge about existing trust relations can be retrieved through a Web service component (Trust Provisioning). On the one side, the management portal itself uses this knowledge, e.g., to suggest new team compositions or activity delegations based on interpersonal trust. On the other side, further third-party collaboration tools could use this knowledge as it is provided in a standardized manner (SOAP and REST-based versions).

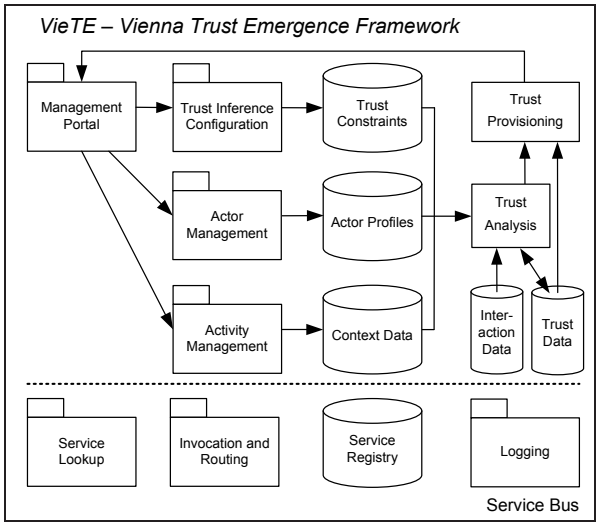


Figure 5. VieTE architecture.

In the following, we discuss in detail the realization of each phase of the MAPE cycle.

A. Monitoring

The Logging component captures interactions during activity execution, such as human communication through services and Web service calls (Listing 1), e.g., through intercepting SOAP calls, and explicit actions undertaken by actors through VieTE's Management Portal, such as the delegation of activities (see Listing 2). Data of both sources are processed, including analysis of faults and interaction correlation to discover request-response patterns [13], and converted to more generic interactions. This generic type describes in detail the type and success of an interaction between two particular actors (Listing 3).

At the end of each run through the monitoring phase, an interaction network is built based on available generic interactions and stored in the interaction database (see Figure 5). We model this network as a directed graph $G_I = (V, E_I)$, whose vertices V represent the actors and multiple edges E_I reflect interactions between them. An edge $e_\imath = (u, v, \imath, ctx)$ is described by the source u of an interaction, the sink v, the generic interaction $\imath$ with its properties, and the interaction context ctx.

```
<ServiceInteraction>
 <clientEndpoint>192.168.0.101</clientEndpoint>
 <messageCorrelationID>000a1460-25ba-...</messageCorrelationID>
 <messageType>Response</messageType>
 <serviceEndpoint>http://www.coin-ip.eu/ss/IMService</serviceEndpoint>
 <eventSourceID>AL-invoke@192.168.0.100</eventSourceID>
 <timeStamp>1207212091812</timeStamp>
</ServiceInteraction>
```

Listing 1. Service interaction log example.

```
<Action xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.coin-ip.eu/ns/action"
xsi:type="CoordinationAction"
ActionURI="http://www.coin-ip.eu/CoordinationAction#6564"
DescribesActivityURI="http://www.coin-ip.eu/Activity#222"
Timestamp="2009-03-05T15:13:21.563Z">
    <ExecutedBy>http://www.coin-ip.eu/Actor#Daniel</ExecutedBy>
    <CoordinationType>
        <DelegateType>Delegate</DelegateType>
    </CoordinationType>
    <ToActor>http://www.coin-ip.eu/Actor#Florian</ToActor>
</Action>
```

Listing 2. Action log example.

```
<GenericInteraction>
 <sender>http://www.coin-ip.eu/Actor#Daniel</sender>
 <receiver>http://www.coin-ip.eu/Actor#Florian</receiver>
 <class>human-human</class>
 <type>Coordination</type>
 <subtype>MeetingOrganization</subtype>
 <numberOfConcernedCoActors>5</numberOfConcernedCoActors>
 <success>true</success>
 <successLevel>3</successLevel>
 <faultLevel>0</faultLevel>
 <faultReason></faultReason>
 <transportService>http://www.coin-ip.eu/ss/IMService</transportService>
 <context>http://www.coin-ip.eu/Activity#222</context>
 <timeStamp>1207212091812</timeStamp>
</GenericInteraction>
```

Listing 3. Generic interaction log example.

B. Analyzing

We understand trust to emerge from and to be highly impacted by the success (or faults) and types of interactions, considered with respect to particular situations described by the introduced context model.

For inferring trust relations, we define a second graph model, the trust network $G_T = (V, E_T)$, where the actors V are the same as in G_I, but the edges $e_\tau = (u, v, \tau, Con)$ reflect the level of direct trust $\tau \in [0, 1]$, emerging from interactions, with respect to the satisfaction of given constraints Con. Algorithm 1 is used to update G_T with the captured G_I, applying the given constraints Con used to weight the influence of interactions on trust calculation.

Algorithm 1 is periodically executed by the system to update the trust network, stored in the trust database (see Figure 5), considering captured interactions in predefined time intervals. The basic mode of operation is as follows: (i) Retrieve G_I built from recent interactions in the previous run of the execution phase. Furthermore get current G_T and configured constraints. (ii) Extract all interactions from G_I between the ordered pair of actors (u, v). (iii) Determine aggregated success score (s_I) and fault score (f_I) for available interactions from u to v. These scores are based on the level of success and fault respectively of an interaction $\imath$, and on the satisfaction of constraints with respect to the interaction context (γ in

Table I
SYMBOL DESCRIPTIONS.

Symbol	Description
α	impact factor of $\Delta\tau$ on τ
Con	set of constraints determining the relevance of $\imath$
ctx	interaction context described by activity structures
$\Delta\tau$	trust obtained from recent interactions captured in the last run of the execution phase
$e_\imath$, e_t	edge in G_I and G_T respectively
f_I	fault score based on a set of $\imath$
γ	relevance of $\imath$ calculated using Con
G_I	interaction network modeled as graph $G_I = (V, E_I)$
G_T	trust network modeled as graph $G_T = (V, E_T)$
$\imath$	generic interaction between actors
s_I	success score based on a set of $\imath$
τ	direct trust $\in [0, 1]$ evolving over time
τ_{rec}	recommended trust from neighbors
u	source vertex of an interaction
v	sink vertex of an interaction

Algorithm 1 Periodic update of G_T with recently captured G_I applying Con

```
/* retrieve G_I = (V, E_I) from the interaction db */
/* retrieve G_T = (V, E_T) from the trust db */
/* retrieve Con from the constraints db */
for each u ∈ V do
    for each v ∈ V do
        if getEdges(E_I, u, v) ≠ ∅ then
            if getEdge(E_T, u, v, Con) = 0 then
                e_t ← createEdge(u, v, 0, Con)
            else
                e_t ← getEdge(E_T, u, v, Con)
            end if
            f_I ← 0, s_I ← 0
            τ ← getTrust(e_t)
            for each e_ι ∈ getEdges(E_I, u, v) do
                ι ← getInteraction(e_ι)
                ctx ← getContext(e_ι)
                γ ← satisfy(Con, ctx)
                s_I ← s_I + successLevel(ι) · γ
                f_I ← f_I + faultLevel(ι) · γ
            end for
            Δτ ← s_I / (s_I + f_I)
            τ ← Δτ · α + τ · (1 − α)
            updateTrust(E_T, e_t, τ)
        end if
    end for
end for
/* save updated G_T in trust db */
/* dispose G_I in interaction db */
```

Equation 1 expresses the relevance of an interaction for given constraints). (iv) Calculate trust ($\Delta\tau$) from u to v only based on recent interactions. (v) Update previous trust τ with $\Delta\tau$ by applying the exponential moving average[1] method using the weighting factor $\alpha \in [0, 1]$. (vi) Update changed trust edge e_t in G_T. (vii) Repeat steps (ii) to (vi) for each ordered pair of actors. (viii) Finally, save G_T, and dispose processed G_I.

The function $satisfy()$ applies Equation 1 to determine the relevance $\gamma \in [0, 1]$ of an interaction by comparing to which extend configured constraints Con match to the interaction context ctx. Each single constraint is set up with a weight. The result of the function $match()$ is either true or false.

$$\gamma = \frac{\sum_{\forall c \in Con} match(c, ctx) \cdot weight(c)}{\sum_{\forall c \in Con} weight(c)} \quad (1)$$

On top of G_T we realize algorithms for deriving recommendations and reputations. Algorithm 2 implements the inference of a collective recommendation trust τ_{rec} from a trustor u to a trustee v, by evaluating all second-hand experiences of the vertices $vertexList$ with respect to set constraints (Con not shown for the sake of brevity). Recommendations from different vertices $w \in vertexList$ may have different impact (e.g., depending on the actors' roles) reflected by $im(w)$. The concept of reputation can be realized in a similar way, but without accounting for the connections to a particular trustor.

Algorithm 2 recommendation(u, v, $vertexList$)

$\tau_{rec} \leftarrow 0$
$sum \leftarrow 0$
for each $w \in vertexList$ **do**
  if $predecessor(w) = u \wedge successor(w) = v$ **then**
    $\tau_{rec} \leftarrow \tau_{rec} + \tau(u, w) \cdot \tau(w, v) \cdot im(w)$
    $sum \leftarrow sum + im(w)$
  end if
end for
return $\frac{\tau_{rec}}{sum}$

The outcome of the analyzing phase is a global view on trust relations between actors. The *Trust Provisioning* Web service interface (see Figure 5) offers the ability to query the periodically updated G_T using, for example, the *Management Portal*. Besides directed trust relations in G_T, recommendations as well as reputations with respect to configured constraints can be dynamically obtained.

C. Planning

Planning collaborations includes the selection of trusted actors for particular activities. Thus, defining the constraints used to infer trust from interactions is part of the planning phase. These constraints are configured for specific trust cases. As an example, consider that trust in humans has to be determined regarding their management skills. Constraints will be set up in order to parameterize the algorithm emphasizing the performance of past organizational activities and management interactions therein.

For defining constraints we integrated the popular Drools[2] engine in VieTE and utilize the Java semantic module. Listing 4 shows an example constraint definition. Furthermore, the relevance of each constraint is weighted (ConWeight). This weight is used by the function $satisfy()$ to determine the degree to which constraints are fulfilled with respect to each interaction's context.

```
<rule−set name="trust_constraints"
   xmlns="http://drools.org/rules"
   xmlns:java="http://drools.org/semantics/java">
   <application−data identifier="results">
    java.util.HashMap
   </application−data>
   <rule name="CheckIfTypeOfActivityIsOrganizational">
    <parameter identifier="context">
     <class>at.ac.tuwien.infosys.viete.InteractionContext</class>
    </parameter>
    <java:condition>
     context.getActivity().getType().equals("organizational")
    </java:condition>
    <java:consequence>
     results.put("RuleActivityTypeIsOrg", ConWeight.MEDIUM);
    </java:consequence>
   </rule>
   <rule name="...">
    ...
   </rule>
</rule−set>
```

Listing 4. Constraint definition example.

D. Executing

In the execution phase the actual collaboration between actors in activities takes place. Every collaboration system requires typical procedures such as escalations that are triggered based on missed deadlines or limited resource availability. VieTE supports these procedures by providing trust relations between affected actors and thus supporting decision making to solve the problems.

V. Discussion

In this section we present an example screenshot of VieTE's *Management Portal* in Figure 6 to demonstrate the application of the VieTE framework in a real world collaboration scenario. We show how the end-user is supported in trust-based selection of actors to perform a specific activity.

In the left frame (Activity Selection) an activity structure is visualized. The details of the selected activity are shown in the lower box, including the name and type, a short description, temporal constraints (deadlines), the current status (pending, running, paused, finished, failed), and assigned resources (e.g., documents).

The right frame (Activity Execution Support) consists of 5 tabs:

- *Actor Evaluation* providing information about the user's personal trust relations to other actors as well as their reputation.
- *Actor Composition* is used for creating new 'mixed' teams, i.e., compositions of humans and services.

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

[2] http://sourceforge.net/projects/drools/

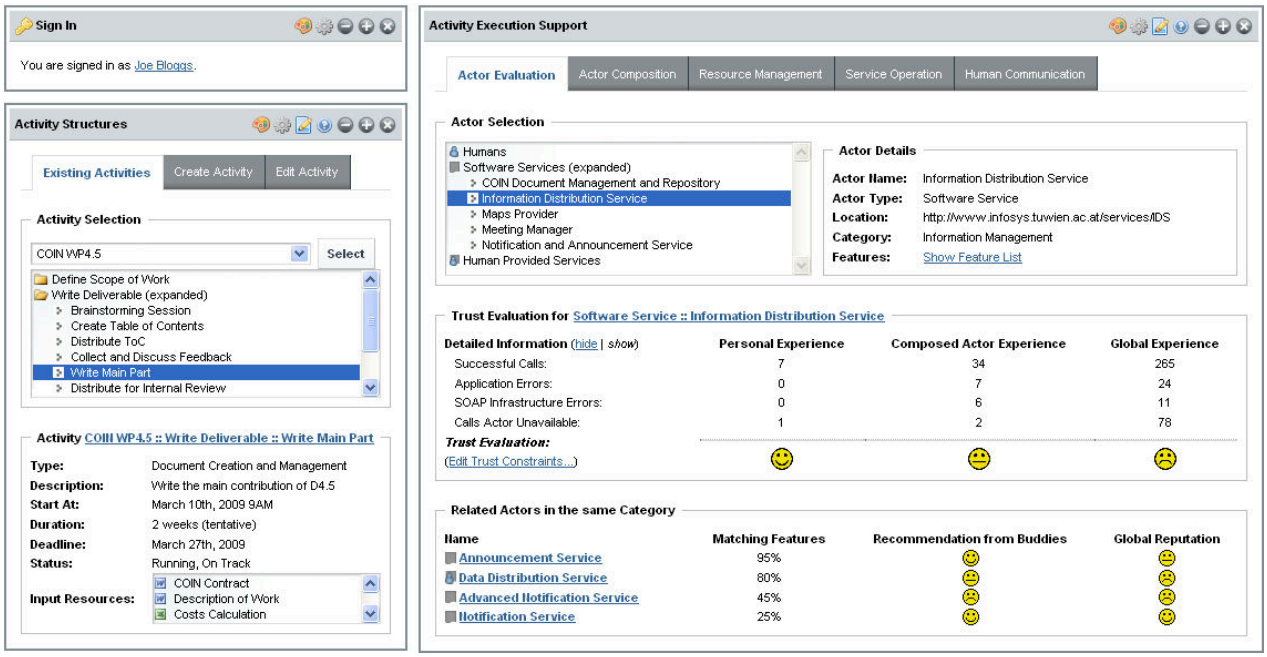


Figure 6. VieTE trust and activity management portal.

- *Resource Management* enables users to manage virtual resources, such as documents, and physical resources, including conference room reservations.
- *Service Operation* provides facilities to dynamically interact with software services by generating custom user interfaces and SOAP messages based on services' operations.
- *Human Communication* provides facilities to dynamically interact with humans by the means of e-mail, instant messaging or text chats.

We assume that the user has certain trust preferences, for example, selecting the most trusted service. (However, at this stage we do not consider trade-off models to account for multiple criteria such as costs versus trust.) Therefore, the top box of the right frame allows the selection of a particular actor to be evaluated. The results of trust evaluation in this co-actor (expressed as emoticons: happy, neutral, sad), based on interaction metrics such as successful calls and availability, is visualized. It is shown that the 'Information Distribution Service' behaves trustworthy for the logged in user (personal evaluation: happy emoticon), however the composed actor experience from the involved activity members is only medium (neutral emoticon). The global experience is largely negative (sad emoticon).

The lower box shows actors with similar features as the currently selected one, for supporting fast replaceability (here: three software services and one Human-Provided service). Furthermore, their recommendations from well-trusted actors ('buddies') and their global reputation is visualized.

VI. Related Work

Recently, trust in service-oriented systems has become a very important research area. Many EU-funded projects such as COIN[3] as well as Master[4] or Spike[5] focus on, for example, trusted collaboration in networked enterprises.

Marsh [14] introduced trust as a computational concept, including a basic definition of trust, the factors it relies on and first concepts to model trust. Based on this work, several definitions of and models for trust have been proposed. Some surveys of trust related to computer science have been performed [7], [8], [15], which outline common concepts of trust, clarify the terminology and show the most popular trust models. From the many existing definitions of trust, we adopt those from [16], [17], which describe that trust relies on previous interactions and collaboration encounters.

Various computational trust models have been introduced ([18], [19]), as well as service reputation models ([20], [21]). However, these models mostly focus human relations or software services only, and are not suitable to manage trust relations in our context-aware mixed systems environments, comprising of collaborating humans *and* services.

Trust in SOA has to be managed and updated in an automated manner. SOA-based infrastructures are typically distributed comprising a large number of available services and huge amounts of interaction logs. A trust management framework for service-oriented environments has been presented in [22], however without considering human actors in SOA.

Context dependent trust was investigated by [7], [8], [14], [17]. Context-aware computing focusing modeling and sensing of context can be found in [23], [24], [25], [26].

[3] http://www.coin-ip.eu
[4] http://www.master-fp7.eu
[5] http://www.spike-project.eu

VII. Conclusion and Further Work

In this paper we presented our approach for the management and inference of trust in mixed service-oriented systems. Interactions in such systems are typically highly dynamic making it impractical to manage trust in a manual manner. At this stage, we focused on the design of the VieTE architecture and a graph-based algorithm to calculate trust. In our future work, we plan to set up and evaluate real scenarios in cross-enterprise collaborations. We will study influence factors on trust inference such as the impact of constraints and fine tuning of the presented algorithm.

Acknowledgment

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

References

[1] 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.

[2] A. Cozzi, S. Farrell, T. Lau, B. A. Smith, C. Drews, J. Lin, B. Stachel, and T. P. Moran, "Activity management as a web service," *IBM Systems Journal*, vol. 45, no. 4, pp. 695–712, 2006.

[3] S. Dustdar, "Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams," *Distributed and Parallel Databases*, vol. 15, no. 1, pp. 45–66, January 2004.

[4] P. Moody, D. Gruen, M. J. Muller, J. C. Tang, and T. P. Moran, "Business activity patterns: A new model for collaborative business applications," *IBM Systems Journal*, vol. 45, no. 4, pp. 683–694, 2006.

[5] F. Skopik, H.-L. Truong, and S. Dustdar, "VieTE - enabling trust emergence in service-oriented collaborative environments," in *International Conference on Web Information Systems and Technologies*, 2009, pp. 471–478.

[6] E. Chang, T. S. Dillon, and F. K. Hussain, *Trust and reputation for service-oriented environments: technologies for building business intelligence and consumer confidence*, Wiley, 2006.

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

[8] 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.

[9] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *International World Wide Web Conference*. New York, NY, USA: ACM, 2004, pp. 403–412.

[10] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions.com community," in *Association for the Advancement of Artificial Intelligence*. AAAI Press / The MIT Press, 2005, pp. 121–126.

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

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

[13] R. Gombotz and S. Dustdar, "On web services workflow mining," in *Business Process Management Workshops*, 2005, pp. 216–228.

[14] S. P. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, University of Stirling, April 1994.

[15] S. Ruohomaa and L. Kutvonen, "Trust management survey," in *iTrust*, ser. LNCS, vol. 3477. Springer, 2005, pp. 77–92.

[16] L. Mui, "Computational models of trust and reputation: Agents, evolutionary games, and social networks," Ph.D. dissertation, Massachusetts Institute of Technology, December 2002.

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

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

[19] G. Theodorakopoulos and J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 318–328, 2006.

[20] S. Kalepu, S. Krishnaswamy, and S. W. Loke, "Reputation = f(user ranking, compliance, verity)," in *International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2004.

[21] E. M. Maximilien and M. P. Singh, "Toward autonomic web services trust and selection," in *International Conference on Service Oriented Computing*. ACM, 2004, pp. 212–221.

[22] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, and K. Nahrstedt, "A trust management framework for service-oriented environments," in *International World Wide Web Conference*, 2009.

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

[24] N. A. Bradley and M. D. Dunlop, "Toward a multidisciplinary model of context to support context-aware computing," *Human-Computer Interactions*, vol. 20, no. 4, pp. 403–446, 2005.

[25] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, "a CAPpella: programming by demonstration of context-aware applications," in *SIGCHI Conference on Human factors in computing systems*. ACM, 2004, pp. 33–40.

[26] S. W. Loke, "Context-aware artifacts: two development approaches," *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 48–53, Apr./Jun. 2006.