Composing Trust Models towards Interoperable Trust Management

Rachid Saadi, Mohammad Ashiqur Rahaman,, Valérie Issarny, Alessandra Toninelli

Composing Trust Models towards Interoperable Trust Management *

Rachid Saadi, Mohammad Ashiqur Rahaman, Valerie Issarny, and Alessandra Toninelli

ARLES Project-Team
INRIA CRI Paris-Rocquencourt, France
{name.surname}@inria.fr

Abstract. Computational trust is a central paradigm in today's Internet as our modern society is increasingly relying upon online transactions and social networks. This is indeed leading to the introduction of various trust management systems and associated trust models, which are customized according to their target applications. However, the heterogeneity of trust models prevents exploiting the trust knowledge acquired in one context in another context although this would be beneficial for the digital, ever-connected environment. This is such an issue that this paper addresses by introducing an approach to achieve interoperability between heterogeneous trust management systems. Specifically, we define a trust meta-model that allows the rigorous specification of trust models as well as their composition. The resulting composite trust models enable heterogeneous trust management systems to interoperate transparently through mediators.

1 Introduction

With people getting increasingly connected virtually, trust management is becoming a central element of today's open distributed digital environment. However, existing trust management systems are customized according to specific application domains, hence implementing different trust models. As a result, it is nearly impossible to exploit established trust relations across systems. While a trust relation holding in one system does not systematically translate into a similar relation in another system, it is still a valuable knowledge, especially if the systems relate to the same application domains (e.g., e-commerce, social network). This is such an issue that we are addressing in this paper.

To the best of our knowledge, little work investigates interoperability between heterogeneous trust models. The closest to our concern is the work of [19], which describes a trust management architecture that enables dealing with a variety of trust metrics and mapping between them. However, the architecture deals with the composition at the level of trust values and do not account for the variety of trust models. In particular, one may want to differentiate between direct trust values and reputation-based ones when composing them. In general, what is needed is a way to formalize heterogeneous trust models and their composition. Such a concern is in particular addressed

* Work supported by EU-funded project FP7-231167 CONNECT and by EU-funded project FP7-256980 NESSoS.

in [9,21], which introduce trust meta-models based on state of the art trust management systems. Nevertheless, little detail is given and the paper does not describe how to exploit the meta-model for composing heterogeneous trust models and this achieve interoperability. Dealing with the heterogeneity of trust models is also investigated in [4,20]. However, the study is for the sake of comparison and further concentrates on reputation-based models. Summarizing, while the literature is increasingly rich of trust models, dealing with their composition remains a challenge.

Towards overcoming the interoperability challenge faced by trust management systems, this paper introduces a comprehensive approach based on the definition of a reference trust meta-model. Specifically, based on the state of the art (Section 2), the trust meta-model formalizes the core entities of trust management systems, i.e., trust roles, metrics, relations and operations (Section 3). The trust meta-model then serves specifying the composition of trust models in terms of mapping rules between roles, from which trust mediators are synthesized (Section 4). Trust mediators transparently implement mapping between respective trust relations and operations of the composed models. While this paper introduces the composition approach from a theoretical perspective, we are currently implementing it as part of the CONNECT project[1] on next generation middleware for interoperability in complex systems of systems (Section 5).

2 Trust Model Definition

As in particular defined in [5]: *i.e.*, *A trustor trusts a trustee with regard to its ability to perform a specific action or to provide a specific service*. Hence, any trust model may basically be defined in terms of the three following elements:

1. *Trust roles* abstract the representative behaviors of stakeholders from the standpoint of trust management, in a way similar to role-based access control model [3].
2. *Trust relations* serve specifying trust relationships holding among stakeholders, and
3. *Trust assessment* define how to compute the trustworthiness of stakeholders.

We further define trust relations and assessment below.

2.1 Trust relations

We identify two types of trust relationships, i.e., *direct* and *indirect*, depending on the number of stakeholders that are involved to build the trust relationship:

Direct trust: A direct trust relationship represents a trust assertion of a subject (i.e., trustor) about another subject (i.e., trustee). It is thus a one-to-one trust relation (denoted *1:1)*) since it defines a direct link from a trustor (**1**) to a trustee (**1**). One-to-one trust relations are maintained locally by trustors and represent the trustors' personal opinion regarding their trustees [10]. For example, a one-to-one relation may represent a belonging relationship (e.g., employees trust their company), a social relationship (e.g., trust among friends), or a profit-driven relationship (e.g., a person trusts a trader for managing its portfolio).

[1] http://connect-forever.eu/

Recommendation-based trust: As opposed to a direct trust relationship, a recommendation-based relationship represents a subject's trustworthiness based on a third party's opinion. This can be either (i) transitive-based or (ii) reputation-based.

Transitive-based trust relations are one-to-many (denoted *1:N*). Such a relation enables a trustor (**1**) to indirectly assess the trustworthiness of an unknown trustee through the recommendations of a group of trustees(**N**). Hence, the computation of 1:N relations results from the concatenation and/or aggregation of many 1:1 trust relations. The concatenation of 1:1 trust relations usually represents a transitive trust path, where each entity can trust unknown entities based on the recommendation of its trustees. Thus, this relationship is built by composing personal trust relations [1,18]. Furthermore, in the case where there exist several trust paths that link the trustor to the recommended trustee, the aggregation can be used to aggregate all given trust recommendations [7].

Reputation-based trust relations are many-to-one (denoted *N:1*) and result from the aggregation of many personal trust relationships having the same trustee. Hence, the N:1 trust relation allows the definition of the reputation of each trustee within the system. Reputation systems may then be divided into two categories depending on whether they are (i) *Centralized* or (ii) *Distributed*. With the former, the reputation of each participant is collected and made publicly available at a centralized server (e.g., eBay, Amazon, Google, [14]). With the latter, reputation is spread throughout the network and each networked entity is responsible to manage the reputation of other entities (e.g., [7,23]).

2.2 Trust Assessment

Trust assessment, i.e., assigning values to trust relationships, relies on the definition of: (i) trust metrics characterizing how trust is measured and (ii) operations for composing trust values.

Trust metrics: Different metrics have been defined to measure trust. This is due to the fact that one trust metric may be more or less suitable to a certain context. Thus, there is no widely recognized way to assign trust values. Some systems assume only binary values. In [24], trust is quantified by qualitative labels (e.g., high trust, low trust etc.). Other solutions represent trust by a numerical range. For instance, this range can be defined by the interval [-1..1] (e.g., [12]), [0..n] (e.g., [1,18]) or [0..1] (e.g., [7]). A trust value can also be described in many dimensions, such as: (Belief, Disbelief, Uncertainty) [7].

In addition, several definitions exist about the semantics of trust metrics. This is for instance illustrated by the meaning of zero and negative values. For example, zero may indicate lack of trust (but not distrust), lack of information, or deep distrust. Negative values, if allowed, usually indicate distrust, but there is a doubt whether distrust is simply trust with a negative sign, or a phenomenon of its own.

Trust operations: We define four main operations for the computation of trust values associated with the trust relations given in Section 2.1 (see table 1): *bootstrapping*, *refreshing*, *aggregation*, and *concatenation*.

The *bootstrapping* operation initializes the *a priori* values of 1:1 and N:1 trust relations. Trust bootstrapping consists of deciding how to initialize trust relations in order to efficiently start the system and also allow newcomers to join the running system

	Bootstrapping	Aggregation	Concatenation	Refreshing
One-to-One (1:1)	X			X
One-to-Many (1:N)		X	X	
Many-to-One (N:1)	X	X		X

Table 1: Trust assessment operations

[16]. Most existing solutions simply initialize trust relation with a fixed value (e.g., 0.5 [6], a uniform Beta probabilistic distribution [8]). Other approaches include among others: initializing existing trust relations according to given peers recommendations [17]; applying a sorting mechanism instead of assigning fixed values [18]; and assessing trustees into different contexts (e.g., fixing a car, babysitting, etc.) and then inferring unknown trust values from known ones of similar or correlate contexts [16,2].

All the solutions dealing with 1:N trust assessment mainly define the *concatenation* and the *aggregation* operations, in order to concatenate and to aggregate trust recommendations by computing the average [18], the minimum or the product [1] of all the intermediary trust values. In the case of Web service composition, some approaches (e.g., [15]) evaluate the recommendation for each service by evaluating its provider, whereas other approaches (e.g., [11]) evaluate the service itself in terms of its previous invocations, performance, reliability, etc. Then, trust is composed and/or aggregated according to the service composition flow (sequence, concurrent, conditional and loop).

Aggregation operations such as Bayesian probability (e.g., [13]) are often used for the assessment of N:1 (reputation-based) trust relations. Trust values are then represented by a beta Probability Density Function [8], which takes binary ratings as inputs (i.e., positive or negative) from all trustors. Thus, the reputation score is refreshed from the previous reputation score and the new rating [14]. The advantage of Bayesian systems is that they provide a theoretically sound basis for computing reputation scores and can also be used to predict future behavior.

Finally, refreshing operations are mainly trigged by trustors to refresh 1:1 and N:1 trust relations, after receiving stakeholders' feedback.

3 Trust Meta-Model

Following the above, we formally define the trust meta-model as: $TM =< \mathbb{R}, \mathbb{L}, \mathbb{M}, \mathbb{O} >$, where $\mathbb{R}$, $\mathbb{L}$, $\mathbb{M}$ and $\mathbb{O}$ are the finite sets of trust roles, relations, metrics and operations.

3.1 Trust Meta-Model Formalization

As detailed below, each set of TM consists of $elements$ where an element can have a $simple\ value$ (e.g., string) or a complex value. A complex value of an element is either an exclusive combination of values (only one of the values) $\vee v$ (e.g., $v_1 \vee v_2 \vee v_3$) or an inclusive combination of values (one or more elements) $\diamond v$ (e.g., $v_1 \wedge v_2 \wedge (v_3 \vee v_4)$) of elements.

Role set $\mathbb{R}$: The role set contains all the roles r played by the stakeholders of the trust model. A role r of $\mathbb{R}$ is simply denoted by its name:

$$r =< \text{name:string} > \tag{1}$$

where: the attribute $name$ of type $string$ represents the name or the identifier of the role[2]. In our meta-model, a stakeholder is represented as a $Subject\ s$, playing a number of roles, r_1, r_2...and r_n, which is denoted as $s \triangleright r_1, r_2...r_n$.

Metric set $\mathbb{M}$: The metric set describes all the trust metrics that can be manipulated by the trust model. A metric is formally denoted as a pair:

$$m =< \text{name:string, type:string} > \tag{2}$$

where: $name$ and $type$ are strings and respectively define the name and the type. The $type$ can be a simple type (e.g., probability([0..1]), label(good, bad), etc.) or a composition of simples ones (e.g., tuple (believe([0..1]), uncertainty([0..1])).

Relation set $\mathbb{L}$: A relation set $\mathbb{L}$ contains all the trust relations that are specified by the trust model. We specifically denote a trust relation as a tuple:

$$\begin{aligned} l =< \text{name:string, ctx:string, type:string, trustor:}\vee r_i\text{, trustee:}\vee r_j\text{, value:}m_k > \\ \text{with } r_i, r_j \in \mathbb{R} \text{ and } m_k \in \mathbb{M} \end{aligned} \tag{3}$$

where: (i) $name$ identifies the relation; (ii) ctx describes the context of the relationship in terms of the application domain (e.g., selling); (iii) $type$ represents the cardinality of the relation and is denoted by one of the following arities: 1:1, 1:N or N:1; (iv) $trustor$ and $trustee$ are roles where a trust relation relates a $trustor$ role with a $trustee$ role; (v) $value$ is an element from the metric set and thus reflects the trust measure given by the trustor to the trustee through this relation. In the above, note that different trustors can establish the same type of relationship with different trustees. Thus, as a trust relation is binary and between a $trustor$ role and a $trustee$, the exclusive combination of roles (e.g., $r_1 \vee r_2 \vee r_3$) is used to describe these elements

Operation set $\mathbb{O}$: The operation set specifies the operations that can be performed over relations by a subject, either to assess the trustworthiness of another subject or to communicate (i.e., request/response) trust values associated with desired subjects (see Figure 1). As defined in Section 2, trust assessment relies on the bootstrapping, aggregation, concatenation and refreshing operations, whereas, the communication of a trust value relies on the request and response operations. An operation is formally denoted as:

$$\begin{aligned} o =< \text{name:string, host:}\vee r_i\text{, type:string, input:}\diamond l_j\text{, output:}\diamond l_k\text{, via:}\diamond l_n\text{, call:}\diamond o > \\ \text{Where } r_i \in \mathbb{R}, l_j, l_k, l_n, \in \mathbb{L}, \text{ and } o \in \mathbb{O} \end{aligned} \tag{4}$$

[2] Note that the name can in particular be specified by an ontological concept that formally describes this role into a given trust ontology although this is not longer discussed in this paper.

where: (i) $name$ identifies uniquely an operation; (ii) $host$ instantiates the role(s) that hosts and executes the operation; (iii) $type$ defines the operation (i.e., request, response, bootstrapping, aggregation, concatenation, and refreshing); (iv) $input$ gives the trust relations that are required to perform an assessment operation or are received by a communication operation; (v) $output$ gives the trust relations that are provided, as the result of either an assessment operation or a communication; (vi) via specifies the trust relationship that should hold with the role with which the communication happens, while its value is *self* in the case of assessment; and (vii) $call$ denotes a continuation (see Figure 1). Note that $input$ and $output$ are complex values, i.e., logical conjunction of one or more relations.

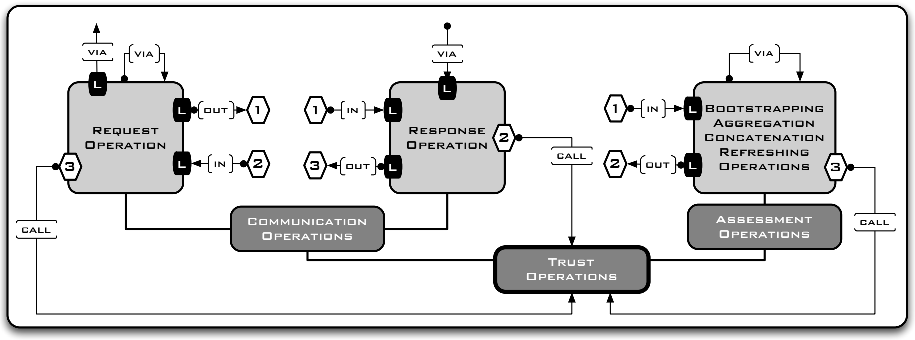

Fig. 1: Operation continuation

Trust graph TG: We associate the definition of a *trust graph* with any trust model TM for the sake of graphical representation. Specifically, the trust graph $TG(\mathbb{R}, \mathbb{E})$ associated with a given TM is a directed graph with the vertices representing the set of roles $\mathbb{R}$ of TM, and the set of edges $\mathbb{E}$ representing the relationship between roles according to $\mathbb{L}$. Hence, each edge is labeled by the referenced relation l from the set of relations $\mathbb{L}$ and the type of that relation, i.e., 1:1, 1:N or N:1.

3.2 Example

We illustrate the expressiveness of our trust meta-model by considering the specification of representative trust models associated with two selling transaction scenarios. Precisely, we introduce the specification of an eBay like centralized trust model (see Table 2) and of a fully distributed one (see Table 3). Both trust models aim at assessing transaction behaviors of sellers.

Figure 2 depicts the trust graphs of both models; the centralized trust model, i.e., TM_C (on the left in the figure), is defined with three roles, i.e., r_S=$Seller$, r_B=$Buyer$, and r_M=$Manager$, whereas the distributed trust model, i.e., TM_D (on the right in the figure), is defined with the unique role r_C=$Customer$, which can be either a seller or a buyer.

Focusing on the specification of TM_C in Table 2 , the roles $Buyer$ and $Seller$ have a direct trust relationship (i.e., l_0) with the $Manager$ that manages the sellers' reputation (i.e., l_3). Thus, any $Buyer$ can: (i) query the $Manager$ about the reputation

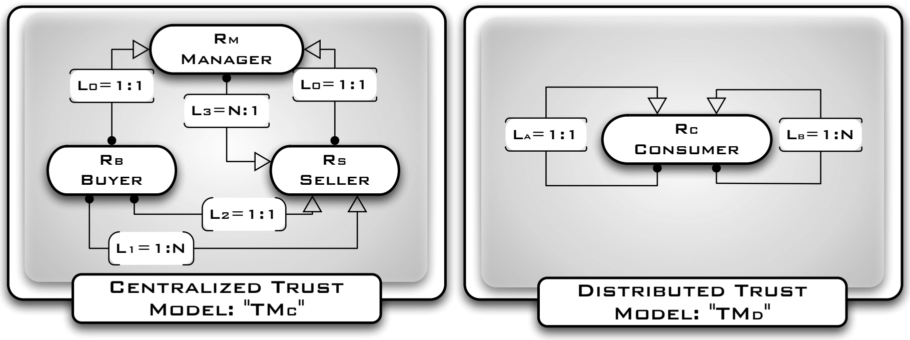

Fig. 2: Trust graphs of the centralized (TM_C) and the distributed (TM_D) trust models.

of a $Seller$ (i.e., l_1), and (ii) provide the $Manager$ with its feedback (i.e., l_2) after a selling transaction. Hence, a $Buyer$ has to perform a request operation (i.e., o_4) to get the reputation of the seller, so that it can compute locally the trustworthiness of the seller (i.e., o_1). After a transaction is completed, a $Buyer$ can provide its feedback to the $Manager$ by triggering a request operation (i.e., o_8). The $Manager$ in turn processes (i.e., o_9) this feedback request to compute and refresh the reputation of the concerned $Seller$ (i.e., o_3).

Role set $\mathbb{R}$	
r_S	= <name="Buyer">
r_B	= <name="Seller">
r_M	= <name="Manager">
Metric set $\mathbb{M}$	
m_0	= <name="Reputation", type="Probability">
m_1	= <name="Recommendation", type="Probability">
m_2	= <name="Rate", type= "Five Semantic labels">
Relation set $\mathbb{L}$	
l_0	= < name="ServerRecommendation", ctx= "Selling", type=1:1, trustor=($r_S \vee r_B$), trustee=r_M, metric=m_1>
l_1	= < name= "SellerTrustworthiness", ctx= "Selling", type=1:N, trustor=r_S, trustee=r_B, metric=m_1 >
l_2	= < name="BuyerFeedback", ctx= "Selling", type=1:1, trustor=r_S, trustee=r_B, metric=m_2 >
l_3	= < name="SellerReputation", ctx= "Selling", type=N:1, trustor=r_B, trustee=r_M, metric=m_0 >
Operation set $\mathbb{O}$	
o_0	= <name="getManagerTrustworthiness", host=($r_S \vee r_B$), type=request, in=l_0, out=l_0 >
o_1	= <name="assessSellerTrustworthiness", host=r_S, type=concatenation, in=($l_0 \wedge l_3$) , out=l_1 >
o_2	= <name= "assessBuyerFeedback", host=r_S, type=update, in=l_2, out=l_2, call=o_8 >
o_3	= <name="setSellerReputation", host=r_M, type=aggregation, in=l_2, out=l_3 >
o_4	= <name="getSellerTrustworthiness", host=r_S,type=request, via=l_0, out=l_1, in=l_3, call=o_1 >
o_5	= <name="getSellerReputation", host=r_M,type=response, in=l_3, out=l_3 >
o_6	= <name="sendSellerReputation", host=r_M,type=response, via=l_0, in=l_1, out=l_3, call=o_5 >
o_7	= <name="getBuyerFeedback", host=r_S,type=request, in=l_2, out=l_2 >
o_8	= <name="sendBuyerFeedback", host=r_S,type=request, via=l_0, out=l_2 >
o_9	= <name="updateSellerReputation", host=r_M,type=response, via=l_0, in=l_2, call= o_3 >

Table 2: Centralized Trust model: TM_C.

Regarding the distributed model TM_D specified in Table 3, the role $Customer$ of the distributed model can maintain a direct trust relationship with other $Customers$

(i.e., l_a) and can then ask trustee $Customers$ to get their recommendation about unknown $Customers$ that are sellers (i.e., l_b). Hence, a $Customer$ can perform a request operation (i.e., o_d) to get a recommendation of an unknown $Customer$ seller, so that the requester $Customer$ can compute locally the trustworthiness of the $Seller$ (i.e., o_b and o_c). After the transaction is completed, the requester $Customer$ can provide its feedback to other $Customers$ by triggering a request operation (i.e., o_f). The recipient $Customer$ can process (i.e., o_h) this feedback to refresh its relationship with the concerned $Seller$ (i.e., o_g) and can also in turn propagate this feedback by calling the o_f.

Role set $\mathbb{R}$	
r_C	= <name="Customer ">
Metric set $\mathbb{M}$	
m_a	= <name="Recommendation", type="Probability">
Relation set $\mathbb{L}$	
l_a	= < name="DirectCustomer Trustworthiness", ctx= "auction", type=1:1, trustor=r_C, trustee=r_C, metric=m_a >
l_b	= < name="TransitiveCustomer Trustworthiness", ctx= "auction", type=1:N, trustor=r_C, trustee=r_C, metric=m_a >
Operation set $\mathbb{O}$	
o_a	= <name="getLocalCustomerTrustworthiness", host=r_C, type=request, in=l_a, out=l_a >
o_b	= <name="assessCustomerTrustworthiness1", host=r_C, type=concatenation, in=$(l_a \wedge (l_a \vee l_b))$, out=l_b, call=o_c >
o_c	= <name="assessCustomerTrustworthiness2", host=r_C, type=aggregation, in=l_b , out=l_b >
o_d	= <name="getRemoteCustomerTrustworthiness", host=r_C, type=request, via=l_a, out=l_b, in=$(l_a \vee l_b)$, call=o_b >
o_e	= <name="sendCustomerTrustworthiness", host=r_C,type=response, via=l_a, in=l_b, out=$(l_a \vee l_b)$, call=$(o_a \vee o_d$ >
o_f	= <name="sendCustomerFeedback", host=r_C,type=request, via=l_a, out=l_a >
o_g	= <name="setCustomerTrustworthiness", host=r_C,type=update, in=l_a, out=l_a, call=o_f >
o_h	= <name="updateCustomerTrustworthiness", host=r_C,type=response, via=l_a, in=l_a, call= o_g >

Table 3: Distributed Trust model: TM_D.

4 Composing Trust Models

Given the specification of trust models, their composition relies on mapping their respective roles so that: (i) the trustworthiness of the various roles can be assessed, (ii) existing trust relations can be queried, and (iii) trust feedbacks can be propagated transparently from one trust model to another. Further, the existing trust relations and operations are extended to relate roles from the composed models, and new assessment operations are required to map trust relations from one model to another. Finally, the resulting mapping and extensions are implemented through mediation [22] so as to make composition transparent to existing systems, which leads us to introduce the corresponding *mediator role*.

Formally, the composition, denoted $\bigoplus$, of two trust models TM_x and TM_y, which introduces the trust model TM_{xy}, is defined as follows:

$$\begin{aligned} TM_{xy} &= TM_x \bigoplus_{\Psi^{xy}} TM_y \\ &= < \mathbb{R}_x, \mathbb{M}_x, \mathbb{L}_x, \mathbb{O}_x > \bigoplus_{\Psi^{xy}} < \mathbb{R}_y, \mathbb{M}_y, \mathbb{L}_y, \mathbb{O}_y > \\ &= \left\langle \begin{aligned} \mathbb{R}_{xy} &= \mathbb{R}_x \cup \mathbb{R}_y \cup \mu\mathbb{R}_{xy} \\ \mathbb{M}_{xy} &= \mathbb{M}_x \cup \mathbb{M}_y \\ \mathbb{L}_{xy} &= \mathbb{L}_x^+ \cup \mathbb{L}_y^+ \\ \mathbb{O}_{xy} &= \mathbb{O}_x^+ \cup \mathbb{O}_y^+ \cup \mu\mathbb{O}_{xy} \end{aligned} \right\rangle \end{aligned} \tag{5}$$

where:

- Ψ^{xy} is the set of mapping rules over roles that enables the composition of TM_x and TM_y;
- $\mu\mathbb{R}_{xy}$ and $\mu\mathbb{O}_{xy}$ are the new sets of mediator roles and mediation operations, respectively;
- ($\mathbb{L}_x^+$ and $\mathbb{L}_y^+$) and ($\mathbb{O}_x^+$ and $\mathbb{O}_y^+$) are the extended relations and operations, respectively.

In the following, we elaborate on the mediation process to generate the sets of mediator roles, and mediation operations (i.e., $\mu\mathbb{R}_{xy}$, and $\mu\mathbb{O}_{xy}$) and extended relations and operations (i.e., $\mathbb{L}_x^+$, $\mathbb{O}_x^+$ $\mathbb{L}_y^+$, $\mathbb{O}_y^+$).

Algorithm 1: Trust_Models_Composition(TM_x, TM_y, Ψ^{xy})

```
Input(s)  : Trust models $TM_x$ and $TM_y$
            The set of Mapping rules $\Psi^{xy}$
Output(s): The trust model composition $TM_{xy} =< \mathbb{R}_{xy}, \mathbb{M}_{xy}, \mathbb{L}_{xy}, \mathbb{O}_{xy} >$
1  begin
       // Initialize trust models sets for composition
2      $\mathbb{L}_x^+ = \mathbb{L}_x$ ; $\mathbb{L}_y^+ = \mathbb{L}_y$
3      $\mathbb{O}_x^+ = \mathbb{O}_x$ ; $\mathbb{O}_y^+ = \mathbb{O}_y$
4      foreach $(\psi_k^{xy} = (\psi_k^{xy} = (r_i : TM_{m=\{x,y\}}) \odot (r_j : TM_{n=\{x,y\},m\neq n})) \in \Psi^{xy})$ do
5          $Relation\_Mediation(r_i, \mathbb{L}_m^+, r_j, \mathbb{L}_n^+, \odot)$
6          if $(\odot ==$ " $\underset{\mu r_k}{\bowtie}$ "$)$ then
7              if $\mu r_k \notin \mu\mathbb{R}_{xy}$ then
8                  $\mu\mathbb{R}_{xy} = \mu\mathbb{R}_{xy} \cup \{\mu r_k\}$
9              $Operation\_Mediation(r_i, \mathbb{L}_m^+, \mathbb{O}_m^+, r_j, \mathbb{L}_n^+, \mathbb{O}_n^+, \mu r_k)$
10     $\mathbb{R}_{xy} = \mathbb{R}_x \cup \mathbb{R}_y \cup \mu\mathbb{R}_{xy}$
11     $\mathbb{M}_{xy} = \mathbb{M}_x \cup \mathbb{M}_y$
12     $\mathbb{L}_{xy} = \mathbb{L}_x^+ \cup \mathbb{L}_y^+$
13     $\mathbb{O}_{xy} = \mathbb{O}_x^+ \cup \mathbb{O}_y^+ \cup \mu\mathbb{O}_{xy}$
14 end
```

4.1 Role Mapping

The mapping of roles from 2 distinct models is explicitly defined through a set of mapping rules defined as follows:

$$\psi_k^{st} = (r_s : TM_s) \odot (r_d : TM_t) \quad (6)$$

where, $\odot$ is asymmetric and maps the source role r_s of TM_s to the target role r_t of TM_t. We further refine $\odot$ into two mapping operators:

- *The See operator*, noted "$\succ$", simply associates a source role with a target role so as to define that the role r_t of TM_t is seen as r_s in TM_t. For instance, in the selling transaction scenarios, $(r_B : TM_C) \succ (r_C : TM_D)$ means that $Buyers$ (i.e., $r_B : TM_C$) of the centralized trust model are seen by the distributed trust model (TM_D) as $Customers$ ($r_C : TM_D$).

– *The $Mimic$ operator*, noted "$\underset{\mu r}{\bowtie}$", specifies that r_s should be able to request trust values of TM_t as if it was r_t. This is practically achieved through the mediator role μr that translates r_s requests into r_t requests. For instance, the rule $(r_C : TM_D) \underset{\mu r}{\bowtie} (r_S : TM_A)$ means that any customer is able to request trust values as if it was a buyer in the centralized trust management system, thanks to the mediation achieved by μr.

The computation of the composition of trust models TM_x and TM_y is detailed in Algorithm 1. The algorithm iterates on mapping rules for each of which it invokes $Relation\_Mediation$ (see line 5) so as to extend relation sets, namely: $\mathbb{L}_x^+$ and $\mathbb{L}_y^+$, (see Section 4.2). Then, according to the definition of $Mimic$ rules, mediator roles (i.e., μr) are added to the set of mediator roles (see lines 7-8), and $Operation\_Mediation$ is invoked so as to perform mediation over the communication operations (see line 9) of the composed trust models (see Section 4.3).

4.2 Relation Mediation

Algorithm 2: Relation_Mediation($r_s, \mathbb{L}_s^+, r_t, \mathbb{L}_t^+, \odot$)

```
    Input(s)  : Roles $r_s$ and $r_t$ ; Relation sets $\mathbb{L}_s^+$ and $\mathbb{L}_t^+$ ;
                Mapping operation $\odot$
    Output(s): The source and the target relation sets: $\mathbb{L}_s^+$ and $\mathbb{L}_t^+$
 1  begin
 2      if $\odot$ = " $\succ$ " then          /* $\Psi^{xy}$ is defined with the "$See$" Operator */
 3          foreach ($l_i \in \mathbb{L}_t^+$) do     /* Find relations with the trustee $r_s$ */
 4              if $l_i.trustee \sqsupset r_t$ then
 5                  $l_i.trustee \xleftarrow{r_t} (r_t \vee r_s)$        /* Add $r_s$ as a trustee */
 6      if $\odot$ = " $\underset{\mu r}{\bowtie}$ " then      /* $\Psi^{xy}$ is defined with the "$Mimic$" Operator */
 7          foreach ($l_i \in \mathbb{L}_s^+$) do     /* Find relations with the trustee $r_s$ */
 8              if $l_i.trustor \sqsupset r_s$ then
 9                  $l_i.trustee \xleftarrow{l_i.trustee} (l_i.trustee \vee \mu r)$     /* Add $\mu r$ as a
                    trustee */
10          foreach ($l_i \in \mathbb{L}_t^+$) do     /* Find relations with the trustor $r_t$ */
11              if $l_i.trustor \sqsupset r_t$ then
12                  $l_i.trustor \xleftarrow{r_t} (r_t \vee \mu r)$        /* Add $\mu r$ as a trustee */
13  end
```

The aim of relation mediation is to extend the trust relations of the original models to roles of the other. More precisely, for any trust relation:
$l =<$ name:string, ctx:string, type:string, trustor:$\vee r_i$, trustee:$\vee r_j$, metric:$m_k >$ of $\mathbb{L}_x$ and $\mathbb{L}_y$ of the composed models TM_x and TM_y, its *trustee* and *trustor* elements are possibly extended to account for mapping between roles.

Algorithm 2 details the corresponding extension where: (i) function $e \sqsupset v$ returns true if v is in e, and (ii) $e \overset{v_i}{\longleftarrow} v_j$ replaces the value v_i in e with the value v_j. As shown in the algorithm, the extension of trust relations depends on the type of the mapping operator. The See operator defines which local trustee (target role r_t) corresponds to the source role (r_s). Therefore, all the relations l_i (from the source trust model) that consider the source role as a trustee ($l_i.trustee \sqsupset r_t$) are extended with the target role (see lines 2-5). The $Mimic$ operator introduces a new mediator role that plays trustees of the source role as a trustee in the source trust model, and plays the target role as a trustor in the target trust model. This leads to the corresponding extension of the trust models relations of $\mathbb{L}_x$ (see lines 7-9) and $\mathbb{L}_y$ (see lines 10-12).

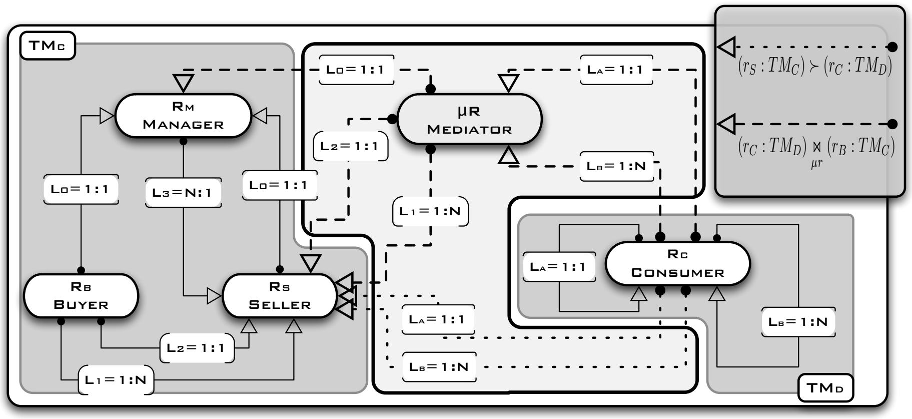

Fig. 3: Trust graph TG_{CD}

Figure 3 depicts the trust graph TG_{CD} resulting from the composition of TM_C and TM_D, while Table 4 details the associated trust roles, metric and relations where new mediator role and extended relations are highlighted in grey. The composition relies on two mapping rules that allow a $Customer$ of TM_D to assess a seller of TM_C. The rule using the See operator represents how sellers are perceived in TM_D, while the second rule using the $Mimic$ operator introduces a mediator role that enables $Costumers$ to request TM_C as $Buyers$. Thus, "$r_B : TM_C \succ r_C : TM_D$" leads to extend the trustee element of l_a and l_b by replacing r_C with $(r_C \vee r_B)$. The mapping rule "$r_C : TM_D \underset{\mu r}{\bowtie} r_S : TM_C$" extends the relations that sink into the role $Customer$ (i.e., l_a and l_b) with the mediator role μr. In addition, all the relations that originate from the role $Buyer$ (i.e., l_0, l_1 and l_2) also originate from the mediator role μr.

Roles set $\mathbb{R}$	
r_S	= <name="Buyer">
r_B	= <name="Seller">
r_M	= <name="Manager">
r_C	= <name="Customer ">
μr	= <name="Customer Mediator">

Metric set $\mathbb{M}$	
m_0	= <name="Reputation", type="Probability">
m_1	= <name="Recommendation", type="Probability">
m_2	= <name="Rate", type= "Five Semantic labels">
m_a	= <name="Recommendation", type="Probability">

Relation set $\mathbb{L}$	
l_0	= < name="ServerRecommendation", ctx= "Selling", type=1:1, trustor=($(r_S \vee \mu r) \vee r_B$), trustee=$r_M$, metric=$m_1$>
l_1	= < name= "SellerTrustworthiness", ctx= "Selling", type=1:N, trustor=($r_S \vee \mu r$), trustee=r_B, metric=m_1 >
l_2	= < name="BuyerFeedback", ctx= "Selling", type=1:1, trustor=($r_S \vee \mu r$), trustee=r_B, metric=m_2 >
l_3	= < name="SellerReputation", ctx= "Selling", type=N:1, trustor=r_B, trustee=r_M, metric=m_0 >
l_a	= < name="DirectCustomer Trustworthiness", ctx= "auction", type=1:1, trustor=r_C, trustee=($(r_C \vee r_B) \vee \mu r$), metric=$m_a$ >
l_b	= < name="TransitiveCustomer Trustworthiness", ctx= "auction", type=1:N, trustor=r_C, trustee=($(r_C \vee r_B) \vee \mu r$), metric=$m_a$ >

Table 4: TM_C and TM_D Composition: Role, Metric, and Relation sets

4.3 Operation Mediation

Operation mediation serves translating request operations from one model into requests in the other model, according to the mappings between roles defined using the *Mimic* operator. More precisely, consider a request operation by r_s for a relation:

<name="l", ctx="c", type="t", trustor="r_s", trustee="tee", metric="v"> of TM_s

where $l \in \mathbb{L}_s$, $tor \in \mathbb{R}_s$, while $tee \in \mathbb{R}_t$ and $r_s{:}TM_s \underset{\mu r}{\bowtie} r_t{:}TM_t$. Then, operation mediation first identifies the matching relations:

<name: string, ctx="c", type: string, trustor="r_t", trustee="tee", metric: m> of TM_t

that should be requested in the target model using a request operation of $\mathbb{O}_t$. Replies are finally normalized using the mediation operation given by $\mu\mathbb{O}_{xy}$ for use in the source trust model. Operation mediation is practically implemented in a transparent way by the mediator that intercepts and then translates r_s requests, as given in Algorithm 3. In the algorithm, the mediator interacts with r_s (see lines 2-4) and r_t (see lines 5-7). Then, the mediator computes the matching relation for each output relation (see lines 11-18) of the reply, where we assume that there is only one such relation (see lines 12-13) and requests its value using the appropriate request operation (see lines 16-18). We further consider that the mediator (μr) embeds a library of mediation functions that translate and normalize heterogeneous trust metrics, which are invoked by mediation operations μo (see lines 12-14). Finally, for each update (i.e., bootstrapping and refreshing) triggered by the response, as specified in the corresponding $call$ element (see lines 19-20), the matching relations is sought in $\mathbb{L}_t$ (see line 23) and its value requested (see lines 25-28).

Figure4 depicts the basic mediation process (left hand side) and its extension with update (right hand side), as performed by the mediator. First, the mediator receives the request in (step 1). Then, it invokes the corresponding request in the target model

Algorithm 3: Operation_Mediation($r_i, \mathbb{L}_m^+, \mathbb{O}_m^+, r_j, \mathbb{L}_n^+, \mathbb{O}_n^+, \mu r_k$)

```
Input(s)  : Source role $r_s$, relation $\mathbb{L}_s^+$ and operation set $\mathbb{O}_s^+$
            Target role $r_t$, relation $\mathbb{L}_t^+$ and operation set $\mathbb{O}_t^+$
            The mediator role $\mu r$
Output(s): The source, the target and the mediation operation sets: $\mathbb{O}_s^+$,
            $\mathbb{O}_t^+$ and $\mu\mathbb{O}_{st}$

1  begin
2      foreach $(o_i \in \mathbb{O}_s^+)$ do          /* Find operation with the host $r_s$ */
3          if $o_i.type = "response" \wedge o_i.via.trustor \sqsupset r_s$ then
4              $o_i.host \xleftarrow{o_i.host} (o_i.host \vee \mu r)$      /* Add $\mu r$ as a host */
5      foreach $(o_i \in \mathbb{O}_t^+)$ do          /* Find relations with the host $r_t$ */
6          if $o_i.type \neq "response" \wedge o_i.host \sqsupset r_t$ then
7              $o_i.host \xleftarrow{r_t} (r_t \vee \mu r)$              /* Add $\mu r$ as a host */
8      foreach $(o_i \in \mathbb{O}_s^+)$ do          /* Find operation with the host $r_s$ */
           // Request mediation
9          if $o_i.type = "response" \wedge o_i.host \sqsupset \mu r$ then
10             if $(o_i.out \neq null)$ then
11                 foreach $l_k \sqsupset o_i.out$ do
                       // Create a new mediated operation $\mu o$
12                     $\mu o.host$=$\mu r$ ; $\mu o.type$="mediation"
                       // Find a similar output relation into $\mathbb{L}_t$
13                     $l^* = findSimilarRelation(l_k, \mathbb{L}_t^+)$
14                     $\mu o.in = l^*$ ; $\mu o.out = l_k$
15                     $\mu\mathbb{O}_{st} = \mathbb{O}_{st} \cup \{\mu o\}$
                       // The relation $l^*$ need to be requested
16                     $o^* = findOperation(type = "request", l^*, \mathbb{O}_t^+)$
17                     $o^*.call \xleftarrow{o^*.call} (o^*.call) \vee \mu o$
18                     $o_i.call \xleftarrow{o_i.call} (o_i.call) \vee o^*$
               // Update mediation
19             foreach $o_k \sqsupset o_i.call$ do
20                 if $o_k.type = "refresh" \vee o_k.type = "booststrap"$ then
21                     foreach $l_p \sqsupset o_k.in$ do
22                         $\mu o.host$=$\mu r$ ; $\mu o.type$="mediation"
23                         $l^* = findSimilarRelation(l_p, \mathbb{L}_t^+)$
24                         $\mu o.in = l_p$ ; $\mu o.out = l^*$
25                         $o^* = findOperation(type = o_k.type, l^*, \mathbb{O}_t^+)$
26                         $\mu o.call = o^*$
27                         $\mu\mathbb{O}_{st} = \mu\mathbb{O}_{st} \cup \{\mu o\}$
28                         $o_i.call \xleftarrow{o_i.call} (o_i.call) \vee \mu o$
29 end
```

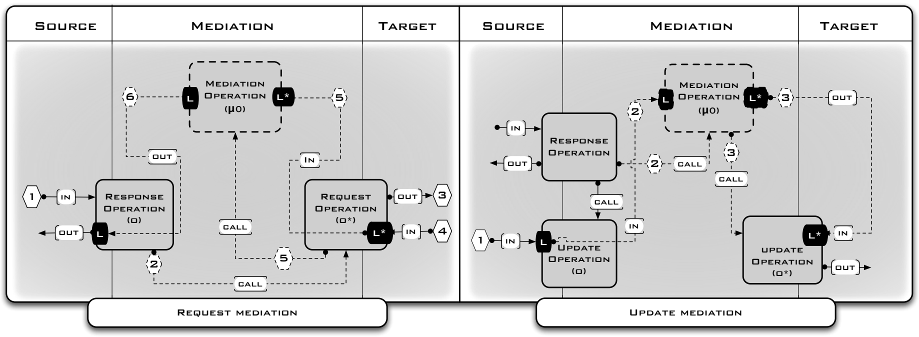

Fig. 4: Operation mediation process

(steps 2 to 4) and upon receipt of the result, it normalizes the value using the mediation operation $\mu\mathbb{O}_{ts}$ (steps 5-6). Finally, the reply *out* is returned. In the case of update (on the figure right hand side), the relation matching the one given as input is sought in the target model using the mediation operation $\mu\mathbb{O}_{st}$ (step 2), leading to invoke the corresponding update operation of the target model (step 3).

As an example, Table 5 gives the operation set $\mathbb{O}_{1,2}$ resulting from the composition of TM_C and TM_D.

Operation set $\mathbb{O}$	
o_0	= $<$name="getManagerTrustworthiness", host=$((r_S \vee \mu r) \vee r_B)$, type=request, in=$l_0$, out=$l_0$ $>$
o_1	= $<$name="assessSellerTrustworthiness", host=$(r_S \vee \mu r)$, type=concatenation, type="product", in=$(l_0 \wedge l_3)$, out=l_1 $>$
o_2	= $<$name= "assessBuyerFeedback", host=$(r_S \vee \mu r)$, type=update, type="rating", in=l_2, out=l_2, call=o_8 $>$
o_3	= $<$name="setSellerReputation", host=r_M, type=aggregation, in=l_2, out=l_3 $>$
o_4	= $<$name="getSellerTrustworthiness", host=r_S,type=request, via=l_0, out=l_1, in=l_3, call=$o_1 \vee \mu o_1$ $>$
o_5	= $<$name="getSellerReputation", host=r_M,type=response, in=l_3, out=l_3 $>$
o_6	= $<$name="sendSellerReputation", host=r_M,type=response, via=l_0, in=l_1, out=l_3, call=o_5 $>$
o_7	= $<$name="getBuyerFeedback", host=$(r_S \vee \mu r)$,type=request, in=l_2, out=l_2, call=μo_2 $>$
o_8	= $<$name="sendBuyerFeedback", host=$(r_S \vee \mu r)$,type=request, via=l_0, out=l_2 $>$
o_9	= $<$name="updateSellerReputation", host=r_M,type=response, via=l_0, in=l_2, call= o_3 $>$
o_a	= $<$name="getLocalCustomerTrustworthiness", host=r_C, type=request, in=l_a, out=l_a $>$
o_b	= $<$name="assessCustomerTrustworthiness1", host=$(r_C \vee \mu r)$, type=concatenation, in=$(l_a \wedge (l_a \vee l_b))$, out=l_b, call=o_c $>$
o_c	= $<$name="assessCustomerTrustworthiness2", host=$(r_C \vee \mu r)$, type=aggregation, , in=l_b , out=l_b $>$
o_d	= $<$name="getRemoteCustomerTrustworthiness", host=r_C,type=request, via=l_a, out=l_b, in=$(l_a \vee l_b)$, call=o_b $>$
o_e	= $<$name="sendCustomerTrustworthiness", host=$(r_C \vee \mu r)$,type=response, via=l_a, in=l_b, out=$(l_a \vee l_b)$, call=$(o_a \vee o_d) \vee o_4 \vee o_7$ $>$
o_f	= $<$name="sendCustomerFeedback", host=r_C,type=request, via=l_a, out=l_a $>$
o_g	= $<$name="setCustomerTrustworthiness", host=$(r_C \vee \mu r)$,type=update, in=l_a, out=l_a, call=o_f $>$
o_h	= $<$name="updateCustomerTrustworthiness", host=$(r_C \vee \mu r)$,type=response, via=l_a, in=l_a, call= $o_g \vee \mu o_3$ $>$
μo_1	= $<$name="Translate$l_1 l_b$", host=μr,type=mediation, in=l_1, out=l_b $>$
μo_2	= $<$name="Translate$l_2 l_a$", host=μr,type=mediation, in=l_2, out=l_a $>$

Table 5: TM_C and TM_D Composition: Operation set

The response operation o_e should be able to assess $Sellers$ of TM_C since its outputs (i.e., l_a and l_b) contain relations that sink into the $Seller$ role (see Table 4). To do so, o_e is extended (see lines 9-18) to enable the mediator role μr (when it performs this operation) to retrieve similar o_e output relations in TM_C, i.e., the relations l_a and l_b that are respectively similar to l_1 and l_2. The operation o_e can hence call o_4 or o_7 to search for l_1 or l_2. Then, as for o_e, the called operations are extended as well, by calling the mediation operations μo_1 and μo_2 to translate respectively l_1 and l_2 into l_b and l_a. Thus, o_e is able to reply the appropriate trust relationships which are interpretable by $Customers$. Moreover, Algorithm 3 (see lines 19-28) enables $Customers$ feedback to be propagated to the $Manager$ of the target model TM_C, so that the reputation of $Sellers$ can be refreshed with the source model feedback. According to the resulting operation set (see Table 5), when the mediator role μr performs the response operation o_h, it calls μo_3 to translate the feedback denoted by the relation l_a into $Buyer$ feedback, I.e., l_2. Then, μo_3 is able to call o_2 with the l_2 to advertise its feedback to TM_C $Manager$.

5 Conclusion

In this paper, we have introduced a trust meta-model as the basis to express and to compose a wide range of trust models. The composition of trust models enables assessing the trustworthiness of stakeholders across heterogeneous trust management systems. Such a composition is specified in terms of mapping rules between roles. Rules are then processed by a set of mediation algorithms to overcome the heterogeneity between the trust metrics, relations and operations associated with the composed trust models. We are currently implementing our approach as part of the *Connect* project[3] where we have defined an XML-based description of the trust meta-model, which we call TMDL (i.e., Trust Model Description Language). Thus, mediators are synthesized on-the-fly given the TMDL description of Trust models.

As future work, we are also considering the implementation of a simulator to a priori assess the behavior of trust composition of given trust models and thus allows fine tuning of the mapping rules. We are also investigating the use of ontologies to specify the semantics of trust model elements and thus possibly infer the mapping rules as well as infer the similarity of trust relations from the semantics.

References

1. A. Abdul-Rahman and S. Hailes. A distributed trust model. In *NSPW: New Security Paradigms Workshop*, pages 48–60, New York, USA, 1997. ACM Press.
2. S. Ahamed, M. Monjur, and M. Islam. CCTB: Context correlation for trust bootstrapping in pervasive environment. In *2008 IET 4th International Conference on Intelligent Environments*, pages 1–8, 2008.
3. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.
4. K. Fullam, T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. Barber, J. Rosenschein, L. Vercouter, and M. Voss. A specification of the Agent Reputation and Trust (ART) testbed. In *Conference on Autonomous agents and multiagent systems*, pages 512–518. ACM, 2005.
5. T. Grandison and M. Sloman. A survey of trust in internet applications. *Communications Surveys & Tutorials, IEEE*, 3(4):2–16, 2009.

[3] http://connect-forever.eu/

6. M. Haque and S. Ahamed. An omnipresent formal trust model (FTM) for pervasive computing environment. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, 2007.
7. A. Jøsang and S. Pope. Semantic constraints for trust transitivity. In *APCCM: 2nd Asia-Pacific conference on Conceptual modelling*, pages 59–68, Newcastle, New South Wales, Australia, 2005. Australian Computer Society, Inc.
8. A. Jsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, pages 17–19, 2002.
9. S. Kaffille and G. Wirtz. Engineering Autonomous Trust-Management Requirements for Software Agents: Requirements and Concepts. *Innovations and Advances in Computer Sciences and Engineering*, pages 483–489, 2010.
10. H. Kautz, B. Selman, and M. Shah. Referral Web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.
11. Y. Kim and K. Doh. Trust Type based Semantic Web Services Assessment and Selection. *Proceedings of ICACT, IEEE Computer*, pages 2048–2053, 2008.
12. S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Scotland, 1994.
13. L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in distributed systems: A bayesian approach. In *Proceedings of the Workshop on Information Technologies and Systems (WITS)*, pages 1–7. Citeseer, 2001.
14. P. Nurmi. A bayesian framework for online reputation systems. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 121–121, Feb. 2006.
15. S. Paradesi, P. Doshi, and S. Swaika. Integrating Behavioral Trust in Web Service Compositions. In *Proceedings of the 2009 IEEE International Conference on Web Services*, pages 453–460. IEEE Computer Society, 2009.
16. D. Quercia, S. Hailes, and L. Capra. TRULLO-local trust bootstrapping for ubiquitous devices. *Proc. of IEEE Mobiquitous*, 2007.
17. A. Rahman and S. Hailes. Supporting trust in virtual communities. *IEEE Hawaii International Conference on System Sciences*, page 6007, 2000.
18. R. Saadi, J. M. Pierson, and L. Brunie. Establishing trust beliefs based on a uniform disposition to trust. In *ACM SAC: Trust, Reputation, Evidence and other Collaboration Know-how track*. ACM Press, 2010.
19. G. Suryanarayana, J. Erenkrantz, S. Hendrickson, and R. Taylor. PACE: an architectural style for trust management in decentralized applications. In *Software Architecture, 2004. WICSA 2004.*, pages 221–230. IEEE, 2004.
20. G. Suryanarayana and R. Taylor. SIFT: A Simulation Framework for Analyzing Decentralized Reputation-based Trust Models. *Technical Report UCI-ISR-07-5*, 2007.
21. L. Vercouter, S. Casare, J. Sichman, and A. Brandao. An experience on reputation models interoperability based on a functional ontology. In *Proceedings of 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2007.
22. G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 2002.
23. R. Zhou, K. Hwang, and M. Cai. Gossiptrust for fast reputation aggregation in peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 1282–1295, 2008.
24. P. R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.