A Context-Aware Trust Model for Service-Oriented Multi-Agent Systems*

Kaiyu Wan[1] and Vasu Alagar[2]

[1] Department of Computer Science, East China Normal University, China
kaiyu.wan@gmail.com
[2] Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada and X'ian Jiaotong-Liverpool University, Suzhou, China
alagar@cs.concordia.ca

Abstract. Service-oriented systems offer the potential for minimizing the development time of business applications within an enterprise while promoting collaborative joint ventures among enterprisers distributed geographically. Service-oriented applications assembled from services obtained from different vendors, sometimes anonymous, should be trustworthy. In this paper we investigate context-aware multi-agent systems (MAS) which can dynamically form coalitions of trusted partners as an effective mechanism to act on behalf of service requestors, find services requested by them, determine trusted services, and provide services to the requestors without violating the privacy of the partners involved in such transactions. The MAS is open with respect to external agents requesting and receiving services, but closed with respect to other activities initiated by external agents. The agents in MAS may have different trust models. We explain how trust models of different agents should be composed into a web of trust for trusted transactions in MAS.

1 Introduction

The concept of service-oriented computing system is not new. Its concepts and characteristics have been detailed in many works [4,23,11]. What is new in this paper is the introduction of *context-aware* MAS as a means of discovering and delivering dependable services to its clients in an open distributed environment. Agents in MAS can dynamically form coalitions of trusted partners as an effective mechanism to act on behalf of service requestors, find services requested by them, determine trusted services, and provide services to the requestors without violating the privacy of the partners involved in such transactions. The MAS is open with respect to external agents requesting and receiving services, but closed with respect to other activities initiated by external agents. The trust models of agents, which may be different, can be combined to generate global trust for agent collaboration in delivering services.

Dependability, recently coined as *trustworthiness* [22], is a composite property. It includes *safety*, *security*, *service availability*, and *reliability* attributes. A service is trustworthy if it is reliable, available without interruption, secure, and safe to use. In order

* This research is supported by a Research Grant from Natural Sciences and Engineering Research Council of Canada(NSERC).

G. Feuerlicht and W. Lamersdorf (Eds.): ICSOC 2008, LNCS 5472, pp. 221–236, 2009.

that an agent recognize and evaluate these attributes in a service, we require it to be context-aware. In agent literature, the BDI (belief, desire, intention) semantics is the semantics for agent actions and interactions. We may interpret the full power of BDI as providing the awareness for an agent. Agents indulging in service-oriented activity may be regarded as software agents, who need not have the full power of BDI semantics, instead its *internal* and its *external* awareness constitute a sufficient semantic basis for its actions.

1.1 Awareness and Context

An agent is aware of its internals such as resources under its care, knowledge of its domain of activity, history of its activities and interactions, and norms (policies) that constrains its actions. We call the internal awareness of an agent as *self-awareness*. The external awareness of an agent includes a knowledge of its environment that may include physical devices, humans and other agents. In order that the system response with respect to a request from its environment be equivalent to a cognitive reaction, the "feel" component of external awareness should be factored in. This is realized by introducing the agent to the *world* with which it has to react. This world consists of the information space of its clients and their behavior. The information space is, in general, *multi-dimensional* where each dimension has information or data with respect to one sort. Aggregating dimensions along with information along the dimensions give rise to a formal definition of context [3,28,29], and justifies formalizing external awareness using contexts. External awareness is known as context-awareness.

Context is *rich* concept. Carnap, in his famous treatise [8] *Meaning and Necessity*, distinguished between the *intension* and *extension* of a natural language statement. Intension is the uttered statement and extension is its truth values in different contexts. As an example, the intension of the statement "The physician in charge of the clinic works from 9 am to 3 pm." is itself, and its extensions are evaluated when the physician's name, the clinic's name, and a time are specified. The world of information to evaluate this statement has three dimensions, namely Physician_name, Clinic_name, and Time. A context is constructed when a value/data is given along each dimension. This meaning of context was tacitly understood and first used in AI, and HCI applications. Since then, context has found several applications in a variety of fields of study within computing, however context itself was represented and interpreted only in an ad-hoc manner. Just to give an overview of its spectrum, see [21,7,14] for logic of context and reasoning in AI, [1] for a tutorial on efforts in formalizing context, [9,10] for HCI applications, [16,17,18] for using context within semantic web for service compositions, [3,28] for high-level languages with context as first class objects, [24,27] for context-aware system architecture and [2,30] for context-aware enforcement of privacy, identity management, and trust. The ability to automatically detect contexts, dynamically de-construct and reconstruct contexts are very relevant to context-aware computing, in particular for mobile and pervasive computing applications. Without a formal representation of context, adaptation and reasoning about the consequences of adaptation is not possible. The formal syntax and the semantic domain for context calculus that we have developed in our research are briefly described in Section 2.

1.2 Trust

Trust, similar to context concept, is also a rich concept. It has a pivotal role in human relationships. McKnight and Chervany [19] have given a typology for classifying trusting behavior in domains such as sociology, psychology, political sciences, and management. A trusting behavior, when expressed in natural language and semantically interpreted through Carnap's intensional logic, becomes an intensional statement. The natural language statements corresponding to these trusting behaviors have a natural intensional logic interpretation. This is the rationale behind the intensional trust model discussed in [30]. In Section 3.3 we show that the six-fold trusting behavior discussed in McKnight and Chervany [19] can be cast within context-aware trust model. Examples 1 and 2 illustrate the necessity to integrate context and trust.

Example 1. *It is remarked in [15] that* inaccuracies in the Wikipedia has been rumored to cause students to fail courses*. Apparently this fault is attributed to a "free for all" approach to the contribution of articles by internet users. Anybody can write an article irrespective of the contributing author's expertise in that area. There is no mechanism to verify either the correctness or completeness of the content in an article. A user who accepts the services offered by such sites will eventually lose her trust in the integrity of the service provider.*

Example 2. *This example is taken from [1], and perhaps the starting point for a formal treatment of context by McCarthy [21]. MYCIN [25] is a computer-based medical consultation system. In particular it advises physicians on treating bacterial infections of the blood. When MYCIN is given the information "the patient has* Chlorae Vibrio*" it recommends two weeks of* tetracycline *treatment and nothing else. What it does not reveal is that there is massive* dehydration *during the course of the treatment. While the administration of* tetracycline *would cure the bacteria, the patient would perish long before that due to diarrhea. Here is an instance where context of usage is not explicitly stated. Although the treatment is in principle correct, it is incomplete. The service is totally* unsafe*.*

If there were to exist a trust model for a system, that model can automatically assign a trust degree based on the completeness and correctness of the information content and offer a personalized rating for every service provided by the system. Completeness and correctness are contextual concepts. Consequently, integrating the trust model with the system awareness of its internal and external contexts, we aim to make the system trustworthy.

1.3 Contributions

We give a rigorous approach to integrate trust model with context-awareness for agents in a MAS. The paper is organized as follows. In Section 2 we briefly outline context formalism. We describe context-aware trust model for service-oriented systems in Section 3. In Section 4 we give a formal context-aware MAS architecture and explain how trusted services are provided by the agents in the absence of a centralized authority. We conclude the paper in Section 5 with a summary of our ongoing research on trustworthy MAS. Appendix 1 lists the formal concepts that we use in MAS architecture.

2 Context Formalism

Context has several working definitions. According to Dey [9] context is *any information that can be used to characterize an entity*, where an entity is *a person, place, or object that is considered relevant to the interaction between user and application*. Context is considered as a *semantic object* by Mrissa etal [18] and this definition is quite close to the formal definition of Wan [29]. Since agents in a MAS need to dynamically construct, assemble, disassemble, modify, and reason with context, a formal definition is necessary. The five most important dimensions from which they should collect information are *who*, *what*, *where*, *when*, and *why*. They respectively provide information on *perception*, to recognize the software component or agent that provides the service or requires the service, *interaction* to determine the type of service, *locality* to determine the location (and its constraints) for providing the service, *timeliness* that specify time bounds for (safe) service delivery, and *purpose* for requested service. It is natural therefore to define a context as a structured typed data: Let $\tau : DIM \rightarrow I$, where $DIM = \{X_1, X_2, \ldots, X_n\}$ is a finite set of dimensions and $I = \{a_1, a_2, \ldots, a_n\}$ is a set of types. The function τ associates a dimension to a type. Let $\tau(X_i) = a_i$, $a_i \in I$. Define a context c as an aggregation of ordered pairs (X_j, v_j), where $X_j \in DIM$, and $v_j \in \tau(X_j)$. The dimensions and the types are suggested by the application of interest.

Example 3. *The vital information to seek an on-line health care service should include the service requestor's credentials. A credential along with location information, date and time can constitute the context. The dimensions are PN (patient name: type string), HC (hospital card number: type integer), LOC (the location where service should be provided: type string), $DATE$ (the date when the service is to be provided: type record).*

2.1 Syntax and Semantics

The context type is determined by the set of dimensions DIM, the types I and the function τ. Our discussion applies to contexts of the same type. The syntax for a context is $[X_i : v_i, \ldots, X_j : v_j]$. The binding between dimensions and the values from the types associated with the dimensions is made explicit. The necessity is that agents should be aware of the dimensions from which the information are perceived. An instance of a context for the type in Example 3 is $[PN : Bob, HC : 123456, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$. For a context c, we write $dim(c)$ to denote the set of dimensions in c, and $val(c)$ to denote the set of typed values in c.

The semantics for contexts is adapted from the set theoretic and relational semantics. we define context operators whose operational semantics are close to the set theoretic and relational operators. Table 1 lists these operators, their precedence and their meanings. By an application of this operational semantics a context expression is evaluated, as in Example 4.

Example 4. *Let $c_1 = [PN : Bob, HC : 123456, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$, $c_2 = [PN : Alice, HC : 456123, LOC : Chicago, DATE : \langle 2008/11/15 \rangle]$, and $c_3 = [PN : Tom, HC : 142536, LOC : Boston, DATE : \langle 2008/10/30 \rangle]$ be three different contexts of the type in Example 3. The expression $c_1 \oplus c_2 \uparrow \{LOC,$*

Table 1. Context Operators and Precedence

operator name	symbol	meaning	precedence
Union	$\sqcup$	Set Union (dimension, value pairs)	3
Intersection	$\sqcap$	Set Intersection (,, ,,)	3
Difference	$\ominus$	Set Difference (,, ,,)	4
Subcontext	$\subseteq$	Subset of dimensions	6
Supcontext	$\supseteq$	Superset of dimensions	6
Override	$\oplus$	Function overwrite	4
Projection	$\downarrow$	Domain Restriction	1
Hiding	$\uparrow$	Range Restriction	1
Undirected Range	$\rightleftharpoons$	Range of simple contexts with same domain	5
Directed Range	$\rightharpoonup$	Range of simple contexts with same domain	5

$DATE\}$ *evaluates to the context* $[PN : Alice, HC : 456123, LOC : Boston, DATE : \langle 2008/10/30\rangle]$. *The result reflects the status wherein* $Alice$ *wants service to be provided at* $Boston$ *on* $2008/10/30$.

Many other properties, such as $dim(c_1 \sqcup c_2) = dim(c_1) \cup dim(c_2)$, can be derived by an application of the semantics[1]. It is sufficient to admit in MAS contexts in which the dimensions are distinct. A context in which a dimension is repeated is equivalent to a set of contexts in each of which the dimensions are distinct [29]. As an example, if Bob wants health care services on two different dates one can construct $c = [PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2008/10/30\rangle, \langle 2009/01/05\rangle\}]$, in which the $DATE$ dimension is repeated. Context c is equivalent to the set with two contexts $c_s = \{[PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2008/10/30\rangle]$, $[PN : Bob, HC : 123456, LOC : Boston, DATE : \{\langle 2009/01/05\rangle\}]$.

Reasoning with Contexts

Let α be a logical expression on some quality attributes that must be verified in a context c. We write $\mathbf{vc}(c, \alpha)$ to denote such a verification condition. The dimensions of a context and the quality attributes that are to be verified in that context may sometime overlap. In that case, the expression α will involve those common dimension names and are treated as free variables. While evaluating α in context c, the dimension names in the logical expression should be bound to some values in their respective typed domains. If sufficient information is not available to evaluate α in context c, the evaluation is postponed, not abandoned. Some simple axioms for verification within a context are the following:

If α is true in a context c_2 then it is true in every sub-context of c_2. That is,

$$\frac{c_1 \subset c_2}{\mathbf{vc}(c_2, \alpha) \Rightarrow \mathbf{vc}(c_1, \alpha)} \tag{1}$$

If α is true in context c then there exists a context c' such that $\mathbf{vc}(c', \mathbf{vc}(c, \alpha))$. That is, for every context c there exists an *outer* context c' from which the verified truths can be observed.

[1] Note that *cup* is a set operator, whereas $\sqcup$ is a context operator.

$$\frac{\mathbf{vc}(c, \alpha)}{\exists c' \bullet \mathbf{vc}(c', \mathbf{vc}(c, \alpha))} \tag{2}$$

Analogous to the Law of Excluded Middle we give the axiom

$$\frac{\mathbf{vc}(c, \alpha \rightarrow \beta), \mathbf{vc}(c, \alpha)}{\mathbf{vc}(c, \beta)} \tag{3}$$

A context c' is a *consistent extension* of context c, written $c' \succeq c$, if $dim(c) \subset dim(c')$, and $\mathbf{vc}(c, \alpha) \Rightarrow \mathbf{vc}(c', \alpha)$. A consistent extension is *monotonic*. Suppose not enough information is available in context c to evaluate α. Since context c' has more (precise) information than context c, $\mathbf{vc}(c', \alpha)$ may have the information to evaluate α.

2.2 Modeling Contact Awareness

A service providing site (agent) must be aware of the different contexts that it is in and the contexts encountered by it. The first kind of awareness is its self-awareness or internal awareness. The second kind of awareness is its external awareness.

Internal Awareness: It is modeled with a view to protect the critical assets of the agent, regulate the service, and optimize resources. It is necessary to abstract in it information regarding it's *clients*, *assets*, *permission policies*, and *obligations*. A client can be an agent acting on behalf of someone, or another entity in the system. In general, they are the subjects who are active in seeking its services. The agent maintains a database (or has access to a database which it can query) of its client categories and identification of clients in each category. Authorization to access the site, get service, and query it can be regulated by identity/password combination or credentials. Assets are the objects that are under its control/ownership. Access to every asset is controlled by the policy (security and privacy) of the institution that the agent represents. For each asset under its control and for each client a permission is determined dynamically, taking into consideration the asset category, the client category, the context of service requirement, and the *purpose* behind the request. An obligation is a policy. There are two kinds of obligations. One kind is "mandatory" that specifies the action (response) that the agent must perform after a user request is fulfilled. For example, if a request for a service is denied, the client shall be informed to get a new authorization, before her session may be terminated. The second kind is obligation set by clients. For example, when a client pays on-line for downloading a software from the site under the agent's control, she may set a time limit for getting a refund on it in case the software fails to work in the configuration set up of the client. Such obligations are usually time constrained. From the above modeling elements, the context type for self-awareness can now be constructed. Assume that $UC = \{UC_1, \ldots, UC_m\}$ is the set of client categories as determined by the site. Let $AC = \{AC_1, \ldots, AC_k\}$ be the set of asset categories which are to be protected. We regard UC_i's and AC_js as dimensions for this context type. Let PC denote the purpose dimension. Assume that the type for UC_i is *string*, the type for AC_i is the set of integers (pointers to files), and the type for PC is the set $\{\text{Accounts}, \text{Clinical}, \text{Registry}\}$. An example of context in this type is $[UC_1 : Alice, AC_2 : EI_1, PC : \text{Clinical}]$. In this context $Alice$ in user category UC_1 is requesting access to the asset EI_1 in category

AC_2 required for a clinic. An obligation is expressed as a logical expression. An example of obligation is $\alpha = onduty(Alice) \wedge 2 \leq deliver\_clinic(Metabolism) \leq 5$, to be interpreted as that $Alice$ is on duty and she must deliver the asset within the time limit $[2, 5]$. In order to enforce the obligation the verification condition $\mathbf{vc}(c, \alpha)$ must be fired at context c.

External Awareness: Contexts that characterize external awareness are constructed automatically from the information gathered from client profile, and client request. The relevant dimensions are LOC, $TIME$, WHO, $WHAT$, $WHERE$, $WHEN$, WHY which correspond respectively to the location from where service is requested, the date/time at which the service request is made, the client role, the nature of service, the location where the service should be provided, the date/time by which the service should be given, and the reason for requesting the service. An example of external context is $c = [LOC$: Montreal, $TIME : d_1, WHO : Manager, WHAT :$ EPR2, $WHEN : d_2, WHY :$ Accounts, $WHERE :$ Boston$]$.

3 An Overview of Trust Model for Service-Oriented Systems

The MAS model that we discuss in Section 4 differs from peer-to-peer servicing systems. The MAS has a middle layer in which agents act as mediators between service requestors and service providers. Service requestors should communicate the quality attributes to the mediators, rather than absolute trust values they expect for a service. The mediators discover the service that satisfies the quality attributes and deliver the service together with a trust value (in a standard ordinal scale) associated with the service to the service requestor. The advantages of this approach include ensuring privacy of service requestors, avoiding the conflicting views on absolute trust values coming from different sources, and guaranteeing the quality of service on behalf of the many service providers whose services might have been composed into the delivered service. We believe that it is necessary to separate trust calculation from trust verification for the following reasons.

Every trust model, regardless of the mechanisms used to measure trust and types of values used to assess and compare trust, will use (1) a trust function, which assigns a trust value on an entity in the context of service request, and (2) enforce trust policies for trust evaluation, and propagation. A system or service is defined to be trustworthy [22] if the four quality attributes *safety*, *security*, *reliability*, and *availability* can be verified in it. If we could formulate these attributes into a logical expression α, then we have to verify $\mathbf{vc}(c, \alpha)$, in every context in which service is provided. This implies that the dimensions used in constructing the service providing context must be sufficiently expressive to include these four quality attributes. The number of dimensions may dramatically increase, and in practice it is not easy to identify all dimensions. Hence, trust calculation, which may be only approximate, must be separated from verification which should be exact. In order to make verification feasible, only those parameters that are recognized as most important for a specific application should be included in the verification condition. As an example, in a movie downloading site the ratings is the most important parameter and it must be verified in the service.

3.1 Choice of Trust Domain

In the literature there exists different conventions for choosing trust values. Regardless of the choice, the trust domain $\mathcal{D}$ which includes all the trust values for an application should be a *complete partial order* [30]. Integers, and real numbers are the natural choices that fit this requirement. If symbols are used, as in stock recommendations or in grading, then we regard it as an *enumerated* type, the order being implicit in the enumeration. As an example, in the enumeration $\{hold, sell\}$, the degree of trust in $sell$ recommendation is higher than the degree of trust in $hold$ recommendation. When the trust domain is either non-numeric or non-simple (such as vectors), it is necessary to define a *metric* on the trust domain to enable the measurement of the distance between two trust values. Such a measurement helps to understand the disparity or closeness between trust values. Towards such a definition of metric here is a simple approach. If $\sigma : \mathcal{D} \rightarrow \omega$ is a total monotone function and $\simeq$ is a partial order on the trust domain then for $d_1, d_2 \in \mathcal{D}$ (1) if $d_1 \simeq d_2$ then $\sigma(d_1) \leq \sigma(d_2)$, and (2) for every chain $d_1, \ldots, d_k$ in $\mathcal{D}$, $\sigma(d_1) \leq \sigma(d_2) \leq \ldots \sigma(d_k)$. This way of metricizing the trust domain is an abstraction of the way that PICS rankings [20,13] are usually interpreted.

3.2 Context-Aware Trust Measurement

We denote the set of entities (subjects and objects in the system) by $\mathcal{E}$, the set of contexts by $\mathcal{C}$, and the set of logical expressions over quality attributes by $\mathcal{Q}$. For some $\alpha \in \mathcal{Q}$, if $\mathbf{vc}(\mathbf{c}, \alpha)$ is true then the trust value in context c should be based upon the quality attributes in α and the information available in context c.

Definition 1. *The function,*

$$\pi : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \times \mathcal{Q} \rightarrow \mathcal{D}$$

associates for $a, b \in \mathcal{E}$, and $c \in \mathcal{C}$ a unique element $d \in \mathcal{D}$, called the trust that a has on b in context c, provided $\mathbf{vc}(\mathbf{c}, \alpha)$ is true.

A service-oriented system is an open distributed system, where sites may have different trust domains and context types. In order to interpret the trust value in one context of one site in a context of another site, a mapping between their trust domains, and another mapping between their context types must be provided. As an example, let us fix the context as common for both sites, say on-line auction, and one site is in Euro zone and another site is in US dollar zone. The sites must have a type conversion function to interpret one another's bidding. In a decentralized MAS, as we explain in Section 4, both type conversions and context homomorphisms can be done by agents.

Reasoning with Context-aware Trust
We fix the context type and trust domain, and give rules for reasoning with trust values.

$$\frac{\mathbf{vc}(\mathbf{c}, \alpha) \wedge \mathbf{vc}(\mathbf{c}, \beta)}{\pi(a, b, c, \alpha \wedge \beta) = \text{maximum}\{\pi(a, b, c, \alpha), \pi(a, b, c, \beta)\}} \tag{4}$$

$$\frac{\mathbf{vc}(\mathbf{c}, \alpha) \wedge \alpha \Rightarrow \beta}{\pi(a, b, c, \beta) = \pi(a, b, c, \alpha)} \tag{5}$$

Corresponding to the axiom 2 we postulate

$$\frac{\exists\beta \bullet \mathbf{vc}(\mathbf{c}', \beta) \wedge \beta \Rightarrow \alpha}{\pi(a, b, c, \alpha) = \pi(a, b, c', \beta)} \tag{6}$$

That is, what is observed from c' should not be invalidated at c'. Corresponding to the subset axiom that if $c_1 \subset c_2$ then $\mathbf{vc}(c_2, \alpha) \Rightarrow \mathbf{vc}(c_1, \alpha)$, we postulate

$$\frac{\mathbf{vc}(\mathbf{c_2}, \alpha) \wedge \mathbf{c_1} \subset \mathbf{c_2}}{\kappa(\pi(a, b, c_1, \alpha) = \pi(a, b, c_2, \alpha))} \tag{7}$$

where κ is the *belief* operator. The trust value, that is believed at one context, can be changed in the same context when more constraints are added to α. In the following equation $\mathbf{vc}(\mathbf{c_1}, \alpha) \wedge \mathbf{vc}(\mathbf{c_2}, \beta)$ is true.

$$\pi(a, b, c_1, \alpha) \geq (\pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{8}$$

$$\pi(a, b, c_2, \beta) \geq \pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{9}$$

$$\text{lub}\{\pi(a, b, c_1, \alpha), \pi(a, b, c_2, \beta)\} \geq \pi(a, b, c_1 \sqcap c_2, \alpha \wedge \beta) \tag{10}$$

where the symbol lub is the *least upper bound* operator. In the next equation assume that $\mathbf{vc}(\mathbf{c_1}, \alpha) \vee \mathbf{vc}(\mathbf{c_2}, \beta)$ is true.

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \pi(a, b, c_1, \alpha) \tag{11}$$

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \pi(a, b, c_2, \beta) \tag{12}$$

$$\pi(a, b, c_1 \sqcup c_2, \alpha \vee \beta) \geq \text{glb}\{\pi(a, b, c_1, \alpha), \pi(a, b, c_2, \beta)\} \tag{13}$$

where the symbol glb is the *greatest lower bound* operator.

3.3 Trust Model vs. Trusting Behavior

We illustrate that the six-fold classification of trusting behavior of an entity a, called *trustor*, on another entity b, called *trustee*, given by McKnight and Chervany [19], can be expressed within context-aware trust model. A trusting behavior can also be associated with one or more of the trustworthiness features *security*, *safety*, *reliability*, and *availability*.

1. *Disposition:* An entity a (client) is naturally inclined to trust b (a vendor). The intention affects directly the actions, without going through any reasoning process. That is, a trusts in *her judgement* in order to trust b, which implies that her judgement on attributes expressed in the expression α is true in every context c the service is offered by b.
2. *Situation:* An entity a trusts b in a particular scenario. As an example, a (consumer) may trust b (broker) in the context c_1 of buying mutual funds but not in the context c_2 of buying stocks. That is, for entity a $\mathbf{vc}(\mathbf{c_1}, \alpha)$ is true and $\mathbf{vc}(\mathbf{c_2}, \beta)$ is false where α is the assertion on the quality attributes for buying mutual funds and β is the assertion on the quality attributes for buying stocks. Axioms 8-13 hold.

3. *Structure:* An entity a trusts the structure (institution) of which the entity b is a member. For example, if b is a chartered bank and a knows the norms of the federal bank of which all chartered banks are affiliates a has the natural inclination to trust b. If b is the mortgage officer in a chartered bank and a is a customer who understands the bank policies on privacy, investments, and mortgage loans, a is inclined to trust b. All the axioms enumerated above are valid. Most importantly trust is *transitive* because of the axioms 3, 5-6.
4. *Belief:* An entity a believes that b is trustworthy. The belief may be based upon factors such as *predictable behavior*, *timeliness*, and *integrity of service*. Consumer a trusts site b because whenever she downloads a software the downloaded software behaves as specified. Trust is calculated in a context, based upon the trust values in the history of contexts encountered by the entity. The belief axiom 7 can be generalized so that trusts at two different contexts, not necessarily subsets, are comparable.
$$\frac{\mathbf{vc}(\mathbf{c_1}, \alpha) \wedge \mathbf{vc}(\mathbf{c_2}, \beta) \wedge \alpha \Rightarrow \beta}{\pi(a, b, c_2, \beta) = \pi(a, b, c_1, \alpha)} \tag{14}$$
5. *Behavior:* An entity a voluntarily depends on entity b. There is no third party involvement here. Consumer a trusts the services provided by a site b because she is either unable to find that service from another site or has not heard anything bad about b's service. Hence consumer a accepts $\mathbf{vc}(\mathbf{c}, \alpha)$ to be true, even when she does not have sufficient information to actually evaluate it.
6. *Intention:* Intention is the result of a desire to reach a goal in a specific context. Once the goal is set in a context an entity a is willing to depend on the services provided by entity b in order to reach her goal. In order to reach the goal, a will try to extend the current context in a consistent manner at each step. Consumer a may choose different b's at different contexts in order to achieve her goal. Formally, let $c_1 \ldots c_n$ be a sequence of contexts such that c_i is a consistent extension of c_{i-1}, $\mathbf{vc}(\mathbf{c_i}, \alpha_\mathbf{i})$ is true, and $\alpha_n \Rightarrow \alpha_{n-1} \ldots \alpha_2 \Rightarrow \alpha_1$. Then we can deduce from the axioms that $pi(a, b, c_1, \alpha_1) \leq pi(a, b, c_2, \alpha_2) \leq \ldots \leq pi(a, b, c_{n-1}, \alpha_{n-1}) \leq pi(a, b, c_n, \alpha_n)$. That is, until the goal is reached the trust values are monotonic non-decreasing, implying that no contradiction is ever introduced.

4 MAS Model of Service

A service delivered in a context c is a function [6] which should satisfy the quality attributes α if $\mathbf{vc}(\mathbf{c}, \alpha)$ is true. To receive and process a service request we model the MAS with four types of agents. The types are defined by the roles and the normative behavior of agents. Many agents of one type may be active in each session of service processing. We write $x : X$ to mean that agent x is of type X.

4.1 Agent Types

The four agent types in the system are UIA, MMA, SPA, and TSA. An agent of type UIA assists users external to the MAS, in formulating service requests, compiling their profiles, deducing the user's information needs by direct communication and history

of transactions, presenting the requests to agents of type MMA, receiving response to a request from MMA agents and communicating them to the respective users, taking corrective actions on behalf of the user, and adapting to the changes in the environment so that it can improve its assistance to the user. An agent of type MMAinteracts with UIA agents to receive service requests and return services, interacts with SPA agents to discover and receive services, and interacts with TSA agents for authentication, certification, trust and context management. An agent of this type SPA provides services in one specific application domain. As an example, *Air Travel* is one specific application within *Transportation* domain. In the MAS model, an SPA agent will provide service in one such specific application domain. Agents of type TSA act as certification authorities, context managers, trust managers, configuration managers, ontologists, vigilante, and auditors. While acting as a vigilante it will monitor events, maintain a history of system evolution, draw inferences, and plan actions to ensure that no agent from outside the AMS has intruded into the system. An important requirement is that all agents in MAS trust the agents who manage their trust information in a secure and manner, assuring integrity and confidentiality. The TSA agents are assumed to have been given sufficient resources to ensure that the services are provided without getting interrupted by external attacks and internal failures. More than one agent may act out the same role, however an agent may act only in conformance with the norms prescribed for its type. An agent may advertise its roles to other agents in MAS and may *subscribe* for their services. An agent may deny service to an agent not subscribed to its services. We use the notation $x : X$ to denote that x is an agent of type X.

4.2 Service Protocol

A service interaction between two agents can happen only if at least one of them is subscribed to the services published by the other. An external client of MAS can subscribe to one or more agents of type UIA. No other MAS agent is visible to an client. Although an external client may subscribe to more than one UIA agent, she cannot interact with more than one UIA agent during a session. A session of a client u, denoted u_s, starts from the instant u contacts one of her UIA agents, say $u_{uia} : UIA$, for service. The session u_s ends when u_{uia} either delivers the service to u or informs u that the requested service is not deliverable. The assumption is that all external clients subscribe prior to starting their sessions.

A subscription is made to an agent when a profile of the requestor is registered with it. The profile includes a specific set of services and a verifiable credential of the requestor. The attributes that make up a profile are determined by the agent to whom the subscription is made. A submitted profile may be modified by the subscriber after the first submission. In turn the agent is obliged to ensure the confidentiality of the registered profiles by announcing the trusted authority in the system with whom the profiles are registered. That is, only a trusted authority agent $taa : TSA$ can maintain the database of profiles collected by an agent u_{uia}. More than one UIA agent may use the services of a taa agent. However, an agent u_{uia} cannot use the services of more than one taa agent.

Protocol Steps

1. *submit service request:* User u contacts one of the agents of type UIA. This micro-step can be just a handshake based on password authorization. Agent u_{uia} from this set presents u with a service template. The user returns an instance of the service template, in which information that can identify the user, a service (domain of service, functionality of service), quality attributes for the service, and contextual information for service delivery are specified. This instance is the specification of the service request, called S_u. The agent u_{uia} sends S_u to its trusted authority taa_{uia}, to the context-management agent $cma : TSA$, and to the matchmaking agent $mma : MMA$[2]. The service specification S_u is structured in such a way each agent that receives S_u can understand and decide which part of S_u is relevant for its task.
2. *matching a profile:* Agent taa_{uia} ignores the information in S_u that is not relevant to its task, and uses the rest of the information to match the profiles in its database. It selects a set of profiles from its database[3] such that in every selected profile the service specified in S_u matches exactly, and the identification information in S_u is either completely or partially matched. If the set of selected profiles is empty it informs u_{uia}, otherwise it sends the selected profile to the authentication agent asa.
3. *authentication of client:* Agent asa has sufficient knowledge in its knowledge base to construct logical assertions from submitted information and reason whether or not a submitted information implies already subscribed information. That is, from the credentials in S_u it forms a logical expression α, from the matching profile of the client received from taa_{uia} it forms another logical expression β and will evaluate $\beta \Rightarrow \alpha$. If it is true, then from the fact that β it can conclude α is true and authenticate the client. It includes the result of authentication in S_u and sends it to u_{uia} and mma.
4. *context construction:* The agent cma has an implementation of context calculus. It extracts the contextual information c_u at which request for service is made and the contextual information c'_u for service delivery from S_u. Context c'_u includes quality attributes. It constructs the contexts in the syntax described in Section 2. It includes the contexts in S_u and send it to mma.
5. *notification of authentication failure:* If u_{uia} receives back S_u a notice of authentication failure, it informs the user u and the security agent asa.
6. *matchmaking process:* Agent mma receives from cma contexts c_u and c'_u. Context c_u is the context in which client u has requested the service, and context c'_u includes the constraints on the quality and other non-functional constraints (such as hardware) to be respected at service delivery to the client. Agent mma has a table of service information gathered from the service providers. For each service provider spa there is an entry in the table, showing the names of services provided by spa, the functionality and quality attribute for each service, and obligation constraints, if any, for each service. We defer a formal discussion on these issues, and assume that mma has the knowledge and resources to match a request against the

[2] For simplicity of exposition assume that there exists only one context management agent, one authentication/security agent, and one match making agent in the system.

[3] The database is a shared resource between trusted authority agent and authentication agent.

services in the table, choose services, and compose them to meet the request. The context-based Service composition approach discussed by Mrissa et al; [18] can be the basis of the knowledge-base of mma. What is needed is to extend this approach so that mma can verify the satisfaction of the composition to the service delivery context c'_u. A brief description of these steps follow. We let $spa_1, \ldots, spa_k$ be the service providers whose services are needed for the composition.

checking credentials of client: Agent mma evaluates $\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})$, where dc_i is a logical expression composed from the institutional policy governing service provision by spa_i. A service provision agent represents some institution and is bound by the policy of that institution. A policy is rule, which being declarative, can be formulated into a logical expression. As an example, the hospital policy regarding physicians to access patient records from the nursing home and for the purpose specified in the context c_u, may have to be evaluated at this stage. In order that u may be permitted to receive the service the expression $\bigwedge_i\{\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})\}$ must be true. If there exists a service provider spa_j for whom $\mathbf{vc}(\mathbf{c_u}, \mathbf{dc_i})$ is false, then that service provider is removed from the list of service providers, and agent mma will contact another spa to get an equivalent service.

service composition: After a successful termination of the previous step, services are composed. Let f denote the composition of services. If α is the logical expression conjoined from the quality attributes (security, trust, reliability, availability, performance) of all spas whose services are composed into f, the assertion claimed by the service providers is that f *satisfies* α (denote as $f\ sat\ \alpha$). That is, the property α is verifiable in the function f. According to Hoare logic, it is sufficient to verify that the post-condition of f implies α.

conformance with expected quality: Agent mma evaluates $\mathbf{vc}(\mathbf{c'_u}, \alpha)$. If it is true, then because the agent has already proved $f\ sat\ \alpha$) it can conclude that α conforms to the quality attributes specified in the service delivery context c', and consequently f is acceptable in context c'_u.

trust calculation: Agent mma sends the list of service providers, the context c'_u and α to the trust management agent tma, which calculates a trust value for mma on each spa_i. The agent tma applies homomorphism and type conversion to normalize the trust values to an ordinal scale, and returns the resulting trust values $\pi(mma, spa_i, c'_u, \alpha)$ to mma.

obligation calculation: Agent mma calculates $\beta = \bigvee_i\{\mathbf{vc}(\mathbf{c'_u}, \mathbf{ob_i})\}$, the obligation constraint.

service provision: From the trust values received, agent mma computes one trust value[4], and sends the service f, the obligation constraint β, along with the computed trust value to agent u_{uia}.

7. *service delivery:* Agent u_{uia} delivers the service f to u, informs u the degree of trust in the service, and enforces the obligation β on behalf of the client u. The enforcement is a firing of the verification condition $\mathbf{vc}(\mathbf{c''_u}, \beta)$, where c''_u is a consistent extension of c'_u.

[4] This can be an *infimum* reflecting the most pessimistic value, or *supremum*, reflecting the most optimistic value, or the average, reflecting a statistical value.

5 Conclusion

In this paper we have given a comprehensive overview of a context-aware trust model for service oriented systems. A service oriented system is envisaged through a multi-agent system, modeled with four agent types. Agents in the MAS can form dynamic coalitions of trusted partners, acting on behalf of service requestors and service providers, and facilitate trustworthy service delivery. The MAS architecture ensures the privacy of participants in a transaction without compromising on the quality of service.

The basic function that computes trust is context-specific, but the actual method used in assessing it is left undefined. We believe that trust formation is a separate issue, and is actively studied by various research groups. Those methods are only empirical. Our approach abstracts the properties of trust, regardless of how they are assessed, and provides a basis for reasoning about trust within a context as well as across different contexts. The important benefits in our approach are: (1) Context is independent of what it references. As a consequence, context-based trust definition captures different kinds information conveyed by events that are observable by agents; (2) Context calculus enables the construction of new contexts from existing contexts, and the logic of contexts, once defined, enables one to reason about the information in the newly constructed context with respect to the information contents in the contexts from which the new one is constructed. As a consequence context-based trust definition is well-suited to handle trust in dynamic networks, in which contexts and their respective information contents may dynamically change independent of each other. Our ongoing work includes a prototype implementation to evaluate the impact of the model on performance attributes, development of a case study on which the theory and performance evaluation are to be applied, and a thorough comparison between our approach and the approach carried out in the semantic web research forums.

References

1. Akman, V., Surav, M.: Steps toward Formalizing Context. AI Magazine, 55–72 (Fall 1996)
2. Alagar, V., Wan, K.: Context Based Enforcement of Authorization for Privacy and Security in Identity Management. In: de Leeuw, E., Fischer Hubner, S., Tseng, J.C., Borking, J. (eds.) IFIP International Federation for Information Processing. Springer Series, vol. 261, pp. 25–38 (2008)
3. Alagar, V.S., Paquet, J., Wan, K.: Intensional programming for agent communication. In: Leite, J., Omicini, A., Torroni, P., Yolum, p. (eds.) DALT 2004. LNCS, vol. 3476, pp. 239–255. Springer, Heidelberg (2005)
4. Barry, D.K.: Web Services and Service-Oriented Architecture: The Savvy Manager's Guide. Morgan Kaufmann Publishers, San Francisco (2003)
5. Bucur, O., Beaune, P., Boissier, O.: Representing Context in an Agent Architecture for Context-Based Decision Making. In: Proceedings of CRR 2005 Workshop on Context Representation and Reasoning (2005)
6. Broy, M.: A Formal Model of Services. ACM Transactions Software Engineering and Methodology 16(1), 1–40 (2007)
7. Buvač, S., Buvač, V.: Mathematics of context. Fundamenta Informaticae 23(3), 263–301 (1995)

8. Carnap, R.: Meaning and Necessity: A study in semantics and modal logic (1947); reprinted by University of Chicago Press (1988)
9. Dey, A.K., Salber, D., Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. Anchor article of a special issue on Human Computer Interaction 16 (2001)
10. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing Journal 5(1), 4–7 (2001)
11. Dijkman, R., Dumas, M.: Service-oriented Design: A multi-viewpoint approach. International Journal on Cooperative Information Systems 13(14) (2004)
12. Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an Agent Communication Language. In: Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM 1994). ACM Press, New York (1994)
13. Grandison, T., Sloman, M.: A Survey of Trust in Internet Applications. IEEE Communications Surveys, 2–16 (2000)
14. Guha, R.V.: Contexts: A Formalization and Some Applications, Ph.d thesis, Stanford University (1995)
15. Korsgaard, T.R., Jensen, C.D.: Reengineering the Wikipedia for Reputation. In: 4th International Workshop on Trust Management (STM 2008), Trondeim, Norway, July 16-17, 2008. Electronic Notes in Theoretical Computer Science (2008), www.elsevier.nl/locate/entcs
16. Maamar, Z., Mostefaoui, S.K., Yahyaoui, H.: oward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions on Knowledge and Data Engineering 17(5), 686–697 (2005)
17. Maamar, Z., Benslimane, D., Narendra, N.C.: What can CONTEXT do for WEB SERVICES? Communications of the ACM 49(12), 98–103 (2006)
18. Mrissa, M., Ghedira, C., Benslimane, D., Maamar, Z., Rosenberg, F., Dustdar, S.: A Context-Based Mediation Approach to Compose Semantic Web Services. ACM Transactions on Internet Technology 8(1), 1–23 (2007)
19. McKnight, D.H., Chervany, N.L.: Trust and distrust definitions: One bite at a time. In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) AA-WS 2000. LNCS (LNAI), vol. 2246, pp. 27–54. Springer, Heidelberg (2001)
20. Miller, J., Resnik, P., Singer, D.: PICS Rating Services and Rating Systems (and Their Machine Readable Descriptions) version 1.1, http://www.w3.org/TR/REC-PICS-services
21. McCarthy, J.: Some expert systems need common sense. In: Pagels, H. (ed.) Computer Culture: The Scientific Intellectual, and Social Impact of Computer. Annals of the New York Academy of Sciences, vol. 426 (1984)
22. Mundie, C., de Vries, P., Haynes, P., Corwine, M.: Trustworthy Computing - Microsoft White Paper. Microsoft Corporation (October 2002)
23. Papazoglou, M.P.: Service-oriented Computing: Concepts, characteristics, and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, Whasington, DC, USA, p. 3. IEEE Computer Society Press, Los Alamitos (2003)
24. Rey, G., Coutaz, J.: The Contextor Infrastructure for Context-Aware Computing. In: Proceedings of 18th ECOOP 2004 Workshop on Component-oriented approach to context-aware systems (2004)
25. Shortcliffe, E.: MYCIN: Computer-based Medical Consultations. Elsivier, New York
26. Toivonen, S., Lenzini, G., Uusitalo, I.: Context-aware Trust Evaluation Functions for Dynamic Reconfigurable Systems. In: Proceedings of WWW 2006, Edinburgh, UK (May 2006)
27. Wan, K., Alagar, V., Paquet, J.: An Architecture for Developing Context-Aware Systems. In: Tzschach, H., Walter, H.K.-G., Waldschmidt, H. (eds.) GI-TCS 1977. LNCS, vol. 48, pp. 48–61. Springer, Heidelberg (1977)

28. Wan, K., Alagar, V.S.: An Intensional Programming Approach to Multi-agent Coordination in a Distributed Network of Agents. In: Baldoni, M., Endriss, U., Omicini, A., Torroni, P. (eds.) DALT 2005. LNCS, vol. 3904, pp. 205–222. Springer, Heidelberg (2006)
29. Wan, K.: Lucx: Lucid Enriched with Context. Ph.d Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada (January 2006)
30. Wan, K., Alagar, V.: An Intensional Functional Model of Trust. Trust Management II. In: Karabulut, Y., Mitchel, J., Hermann, P., Jensen, C.D. (eds.) IFIP 2008, pp. 69–85. Springer, Heidelberg (2008)

Appendix – Type Conversion and Homomorphism

Homomorphism: In a distributed MAS, agents are most likely to have different trust domains. In order that trust values across agents can be compared and global trust computed it is necessary that a homomorphism ϕ, defined below, be a continuous function on the partial order trust chains.

$$\phi : (\mathcal{D}, \simeq, \sigma) \rightarrow (\mathcal{D}', \simeq', \sigma')$$

- for $d_1, d_2 \in \mathcal{D}$, $\phi(d_1 \simeq d_2) = \phi(d_1) \simeq' \phi(d_2)$
- $\phi(\sigma(d_1, d_2)) = \sigma'(\phi(d_1), \phi(d_2))$

If a homomorphism, as defined above exists, then trust value from one domain can be converted to a trust value in another domain. Let $\pi : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \rightarrow \mathcal{D}$ and $\pi' : \mathcal{E} \times \mathcal{E} \times \mathcal{C} \rightarrow \mathcal{D}'$ be two functions that give the trust values on the trust domains $\mathcal{D}$ and $\mathcal{D}'$. Then $\phi \circ \pi = \pi'$.

Context Type Conversion:

[DIM and I are same:] Let $\tau : DIM \rightarrow I$ and $\tau' : DIM \rightarrow I$ be two context types, $\tau \neq \tau'$. Let $DIM_1 \subset DIM$, and $I_1 \subset I$, such that $\tau = \tau'$ when restricted to $DIM_1 \rightarrow I_1$. Let $DIM_2 = DIM \setminus DIM_1$, and $I_2 = I \setminus I_1$. Then for every $X_i \in DIM_2$, $\tau(X_i) = a_i$, $\tau'(X_i) = a_j$, $a_i, a_j \in I_2$, and $a_i \neq a_j$. However, if the two types a_i and a_j have a common super-type b_{ij}, in the sense defined below, then we can replace values of types a_i and a_j with a value in b_{ij}.

Definition 2. *If there exists a type b_{ij} and two maps $\iota_i : a_j \rightarrow b_{ij}$ and $\iota_j : a_i \rightarrow b_{ij}$ such that $\iota_j \circ \tau = \iota_i \circ \tau'$ then the type b_{ij} can be used for the dimension X_i. Contexts $c_1 = [X_i : x]$, $x \in a_i$ and $c_2 = [X_i : y]$, $y \in a_j$ are type convertible to $[X_i, z]$ because there exists $z \in b_{ij}$ such that $z = \iota_j(x) = \iota_i(y)$.*

[(DIM,I), (DIM,I') are two different pairs:] The context types are $\tau : DIM \rightarrow I$, and $\tau' : DIM \rightarrow I'$. If there exists a map $\phi : I \rightarrow I'$ such that $\phi \circ \tau = \tau'$ then contexts of type τ can be converted to contexts of type τ'.

[DIM and DIM' are two different sets of dimensions:] Let $\tau : DIM \rightarrow I$ and $\tau' : DIM' \rightarrow I'$ be two context types. If there exists $D \subset DIM$ and $D' \subset DIM'$ such that $\mid D \mid = \mid D' \mid$, and for each $d \in D$ there exists a unique $d' \in D'$ such that $\tau(d) = \tau'(d')$ then the dimension pairs (d, d') are alias to each other. Hence, from the context type τ we can project out the sub-contexts on dimensions in the set D, and then rename every context in $d \in D$ with its alias in D'. In practice, this is much easier if the ontologist, an agent in MAS, has the alias table and helps the cma agents to extract contexts that suit the dimensions gathered by its clients.