A FORMAL MODEL FOR TRUST LIFECYCLE MANAGEMENT

Waleed Wagealla[1], Marco Carbone[2], Colin English[1], Sotirios Terzis[1], and Paddy Nixon[1]

[1] Department of Computer and Information Sciences,
University of Strathclyde in Glasgow,
26 Richmond Street, Glasgow, G1 1XH United Kingdom
firstname.surname@cis.strath.co.uk
[2] BRICS, University of Aarhus,
Ny Munkegade b.540, 8000 Aarhus C, Denmark
carbonem@brics.dk

Abstract. The rapid development of collaborative environments over the internet has highlighted new concerns over security and trust in such global computing systems. The global computing infrastructure poses an issue of uncertainty about the potential collaborators. Reaching a trusting decision in such environments encompasses both risk and trust assessments. While much work has been done in terms of modelling trust, the investigation of the management of trust lifecycle issues with consideration of both trust and risk is less examined. Our previous work addressed the dynamic aspects of trust lifecycle with a consideration of trust formation, exploitation, and evolution. In this paper we provide an approach for formalizing these aspects. As part of the formalization of the trust lifecycle, we introduce a notion of *attraction* to model the effect of new pieces of evidence on our opinion. The formalization described in this paper constitutes the basis of ongoing work to investigate the properties of the model.

1 Introduction

Increasing interest in collaborative applications for accomplishing difficult distributed tasks in the global computing context (Internet and other large networks) has led to an increased awareness of the security issues in such a dynamic and open environment [5], [4]. It is becoming widely acknowledged that traditional security measures fail to provide the necessary flexibility for interactions between mobile autonomous entities, due to statically defined security policies and capabilities. The global computing setting precludes reliance on some central control authority, which in traditional systems would have full knowledge of all the entities within the system. One approach that has been suggested to help alleviate these problems is the concept of trust [7], which in analogous situations in human society has proved to be effective in allowing unknown entities to learn to determine how each other are likely to behave based on evidence that is available for evaluation. In the SECURE project[9], we are providing an approach that considers risk and trust to secure collaborations between entities. At the heart of the SECURE approach lies the formal trust model [2] and risk model [1].

In a global collaboration environment, mobile entities form ad-hoc collaborations in order to achieve their goals. These entities might represent persons, devices, or their software agents. Every entity that needs to make trusting decisions is defined in our model as a principal in a set P. A collaboration involves a number of actions belonging to the set A, the set of all possible actions. Although a collaboration in the general case might involve multiple actions and principals, in this paper we restrict ourselves to single action collaborations between two principals. We call such collaborations simple. Thus, a simple collaboration is defined as follows:

Definition 1 (Simple Collaboration). *Given a set P of principals and a set of actions A, a simple collaboration C between p_i and p_e elements of $\mathcal{P}$ is defined as a triple (act, p_i, p_e) , where*

- ▷ *p_e is the executor of the action,*
- ▷ *p_i is the initiator of the action,*
- ▷ *act denotes the action (actions could be parameterized) that p_i wants to perform and is an element of A, the set of actions.*

The starting supposition is that one of the two collaborators (p_i) sends a request to p_e asking for permission to perform the action *act*. Since the environment of collaboration is characterized by a high level of uncertainty about the identity and trustworthiness of the collaborators, a trusting decision needs to be made. In such a situation both trust and risk assessment play a crucial role in establishing collaboration, in the sense that no trusting decision is required if there is no risk involved in executing the action. This is the reason for having trust and risk models underpinning the SECURE project.

The structure of the paper is as follows: section 2 describes the formal model of the trust-based decision maker with its components. Section 3 discusses the trust policy languages. We conclude and state the future work in section 4.

2 Trust-based decision maker

The approach we are taking in the SECURE project is to develop a dynamic model of trust to provide entities with the ability to collaborate and make security related decisions autonomously. A trusting decision will be reached by weighing the trust information of the principal and the risk assessment for the requested interaction. The cyclic relationship between trust and risk implies that on one hand, trust is a parameter in the risk assessment, while on the other hand risk is a parameter in the trust evolution process.

Figure 1 illustrates the structure of the decision maker that incorporates four components: *a trust engine, a risk evaluator, a trust lifecycle manager,* and *an evidence repository*. For a trust-based decision to be made, the trust engine supplies the risk evaluator with the trust information that would help in reasoning about the risk involved in the collaboration. For example, a trustworthy principal behaves in a way that increases the probability of high benefits and reduces the probability of high costs. The decision is based on the risk aversion of the deciding principal and the expected costs/benefits of the interaction as they are dictated by risk analysis. The trust information , in the form

of evidence from interactions, collected and processed by the trust lifecycle manager will be fed into the risk evaluator and the trust engine to be used in future interactions. Trust information or values may be stored in memory to represent historical information on the behavioural patterns of specific entities. The functionalities of these components are further discussed in the following subsections.

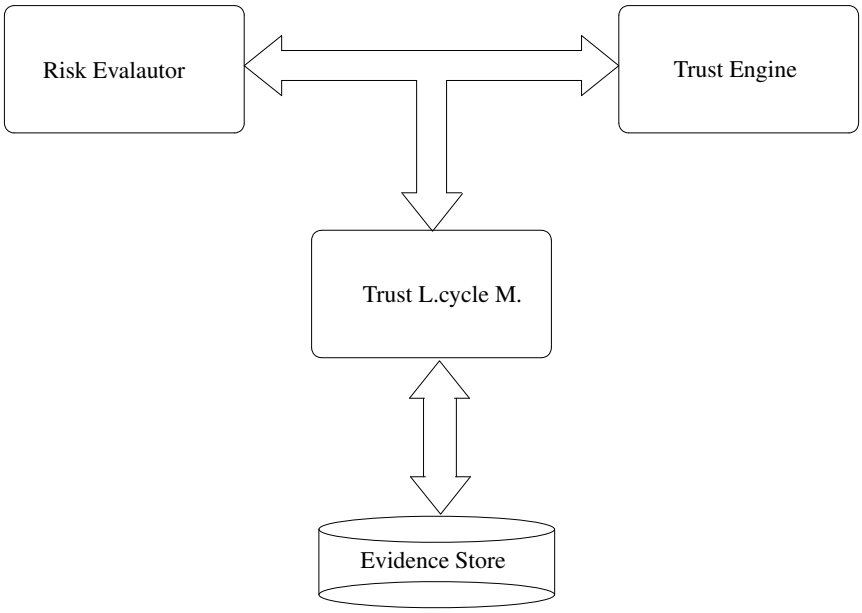

Fig. 1. The trust-based decision maker

2.1 Trust Engine

The aims of the trust engine are to update trust information based on the accumulation of evidence and to supply the risk evaluator with the required trust information for handling requests and reaching decisions.

In the global computing setting, the set of active principals is very large. Therefore, every principal has its own trust values on a finite set of other principals and associates *unknown* to the rest (*unknown* is interpreted as having no evidence for trust or distrust).

In the trust engine [2], the global trust of the system is expressed as a function (m) mapping principals ($\mathcal{P}$) to a function from principals to trust information ($\mathcal{T}$). Formally we have:

$$m : \mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathcal{T}$$

meaning that m applied to a principal a and then to a principal b would return the trust value $m(a)(b) \in \mathcal{T}$ expressing a's trust in b. The set $\mathcal{T}$ is a set of trust values, whose

elements represent degrees of trust. The set $\mathcal{T}$ has two orderings $\preceq$ and $\sqsubseteq$ such that $(\mathrm{T}, \leq)$ is a complete lattice and $(\mathrm{T}, \sqsubseteq)$ is a complete partial order (c.p.o.) with a bottom element. The two orderings ($\preceq$ and $\sqsubseteq$) that defined on the set $\mathcal{T}$ are:

The first ordering ($\preceq$) is the trust ordering, that allows the qualitative comparison of trust values. For example, in a complete lattice $(\mathrm{T}, \leq)$ that expresses a set of trust labels (*low,medium, high*) the value *high* reflects "more trust" in comparison to the value *medium*.

The second ordering ($\sqsubseteq$), the trust information ordering, represents the quantity of available trust information for a principal. In essence this enables the comparison of the precision of trust information.

Following these two orderings, we can construct the triple $(\mathrm{T}, \leq, \sqsubseteq)$ starting with a complete lattice, $(D, \leq)$, and considering the set of intervals of type $[d_0, d_1]$ over D. The $\preceq$ ordering on this set of intervals allows the construction of a complete lattice of intervals which can be allocated as trust values. For example, the interval [*medium, high*] represents higher trust than the interval [*low, medium*].

The $\sqsubseteq$ ordering considers the intervals' width, which can be thought of as representing the amount of uncertainty. For example, the interval [*low,medium*] represents less uncertainty in comparison to the interval [*low, high*]. In fact, the interval expresses that the exact trust label could be either *low* or *high*.

Each principal has a trust policy that expresses how the principal plans to compute its own trust information (for examples of policies see section 3 of this paper). The global trust function *m* will be computed from the collection of all the policies (one for each principal).

There are two operations defined in the trust engine: *update* and *trust*. The *update* method modifies the state of the engine with new observation, and the *trust* method returns a trust value for a principal. Local policies compute the trustworthiness of the principal.

2.2 Risk Evaluator

The function of the risk evaluator is to reason about the risk involved in the interactions between principals. The risk evaluator helps in reaching decision under the circumstances of uncertainty. For this reason the whole process of the risk evaluation is based on investigating the likelihood of potential costs or benefits of the possible outcomes.

Each action is associated to a set of possible *outcomes*. Each outcome has an associated subjective family of cost-probability density functions (*cost-pdfs*)[1]. The cost-pdfs represent the range of possible costs and benefits that may be incurred by the user for each outcome, reflecting variations in principal behaviour. Actions might have parameters associated with them. In this case, the action parameter may affect the values of the costs/benefits.

The trust information that is provided by the trust engine selects one cost-pdf from the family of cost-pdfs for each outcome. The selected cost-pdf for each of the independent outcomes are combined and analyzed according to a security policy, to facilitate a decision on the requested action. The decisions need not be binary, in the sense of "accept or reject", but may also encode some other decisions, *i.e.* asking for more information.

The approach we are taking in the SECURE project is to incorporate trust and risk for trust-based decision making for collaboration between entities. Reasoning about risk and trust indicates the mapping relationship between users' trustworthiness and users' profile of expected behaviour (cost-pdfs).

2.3 Trust Lifecycle Manager

The dynamic aspects of the model such as how trust is formed, how trust evolves over time due to available information and how trust can be exploited are collectively referred to as the trust lifecycle. Therefore, the trust lifecycle manager evaluates trust information and reflects that on the level of principals' trustworthiness. The processes of trust formation and trust evolution are slightly different. Formation differs from evolution in that additional evidence might be actively sought for formation, while evolution refers to the process of changing one's estimation of trust based on gathered trust information from interactions.

There are two main sources of trust information (evidence): *observations* and *recommendations*. Personal observations are gathered after each collaboration is complete. Personal observations of the entity's behaviour are essential for the subjective evaluation of trustworthiness; therefore the outcome of collaborations is recorded and stored as evidence for future collaborations. Recommendations from trusted third parties provide the possibility for trust regarding unknown entities to be propagated.

It is important to state that we would expect personal observations to influence trust to a greater degree than recommendation, therefore it is important to weight the evidence dependent on the source of the information. Since trust is a subjective notion, our own trust information has more merit in the sense that we have first hand knowledge of the whole circumstances of the interaction. Recommendations, however, are passed on as a trust value, which conceals much of the supporting information. The process of recommendation becomes more important in cases where we have no personal interactions with the entity in question.

In [6], we introduced the notion of attraction. The attraction represents the effect of a new piece of evidence on the current trust value. The new evidence (either from observations or recommendations) has some influence on our opinion, in the sense that it "attracts" our opinion towards it.

Definition 2 (Attraction). *An attraction att is a pair* (τ, ι) *where* τ *is the direction of the attraction and* ι *is the power (strength) of the attraction.*

The attraction operates by using the two orderings: trust ordering ($\preceq$) and information ordering ($\sqsubseteq$). Accordingly, we have that:

- ▷ The direction of the attraction, τ, is determined by the trust ordering. Relative positioning of the two trust values according to the trust ordering, i.e. trust positive, trust neutral and trust negative. For example, if T_{curr} = [*low, medium*] and T_{evd} = [*medium, high*] then the trust value from evidence is trust positive.
- ▷ The power (strength) of the attraction, ι, is determined by the information ordering. Relative strength of the two trust values is determined by their positioning on the information ordering c.p.o. For example, if T_{curr}=[*low,medium*] and T_{evd}=[*medium,high*] then the trust value from evidence is conflicting.

The power (strength) has the following properties:

- ▷ The stronger the new piece of evidence, the stronger the attraction (τ and ι) is.
- ▷ The stronger our opinion the weaker the attraction.
- ▷ A supporting evidence reinforces our opinion.
- ▷ A contradicting evidence undermines our opinion.

Before describing the application of the notion of attraction and its two functions to observations and recommendations, we will discuss the evaluation of observations and recommendations.

Observation. Observation is defined as the observed cost-benefit for each outcome upon completion of an interaction (*ocb-outcome*). The evaluation function takes place after completing an interaction and obtaining some observation. Formally the evaluation of observation is achieved by the function `eval()`. This function takes as arguments the current trust value (T_{curr}) and the observed costs-benefits of the outcomes and returns a trust value.

As described above in the risk evaluator, the outcomes are associated with their potential cost and likelihood. Incorporating the trust model and the risk model allows us to choose the best cost-pdf for the outcomes. By reaching a decision, a prediction of the expected cost/benefit of the outcomes will be made. The evaluation function determines the level of trust in the principal based on the one piece of evidence, by comparing the current trust value used to initiate the collaboration with the observed cost/benefit (*ocb-outcome*). The evaluation function must therefore be able to determine the correct trust value which corresponds to the ocb-outcome, to enable the determination of the relevant trust value. This is achieved through the use of a reversible mapping from cost/benefit to trust values.

The evaluation function $\texttt{eval}_p^{act} : \mathcal{T} \times \mathit{Outcomes} \rightarrow \mathcal{T}$, where Outcomes is the set of possible outcomes and T is the set of trust values. The function takes a pair (ocb-outcome, T_{curr}), where ocb-outcome represents the observed cost/benefit and the T_{curr} represent the current trust value that was in use before establishing the interaction. The evaluation function provides an insight into which cost-pdfs would have been the more accurate predictors of the outcome of the interaction (i.e. which set of cost-pdfs would have provided the highest likelihood for the observed costs-benefits). This evaluation function produces a trust value (i.e. the trust value that would have picked this set of cost-pdfs). The evaluation function *eval()* could be defined as,

$$\texttt{eval}_p^{act} = \mathit{eval}(T_{curr}, \textit{ ocb-outcome,}C) = \mathit{eval}(T_{curr}, \textit{ ocb-outcome}, act, p_e, p_i)$$

where *act* is the action and p_e and p_i are the principals. Following this approach evidence might be stored with two indexes identifying the action and the principal who performed the action. This allows us to look up the history of evidence by either the principal or the associated action.

Recommendation. Recommendations are defined as,

$$\mathit{rec} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathcal{T}$$

For example if $a, b \in \mathcal{P}$ ($\mathcal{P}$ is the set of principals) we can say that $rec(a)(b)$ is a's trust value for b. We seek recommendation for further information when the amount of observations is insufficient. Accordingly, we request recommendations, with respect to the principal in question, from trusted third parties. Trusted third parties pass on their opinion on the principal in question as a trust value. This trust value summarizes their own observations. Having recommendations based on third parties' observations prevent us from receiving second hand recommendations.

Assumption 1 (Uniform Behaviour).

We only consider recommendations from principals that observe uniform behaviour to us, In the sense that we observe similar profile of cost/benefit probability mass.

The driving force behind this assumption is that we only consider recommendations provide information useful to us. Unless we observe similar behaviour then a recommendation is misleading.

There is an issue of discounting recommendations. The current state of our formalization does not incorporate a notion of discounting recommendations. We are considering this issue in our agenda of future work. The discounting will change the trust value before its evaluation otherwise the approach is the same.

The way we deal with recommendations is similar to observations by calculating their respective trust values' attraction. Each piece of evidence (either from recommendations observations) has its own attraction on our trust value.The core step of the evolution of trust values is the application of the following two attraction-functions to the new piece of evidence.

As said above, the attraction-functions correspond to the notion of the two orderings of the trust engine (*trust ordering* and *information ordering*). Therefore, we have two attraction-functions: trust function (τ-function) and information function (ι-function). These two functions take as arguments the current trust value T_{curr} and the trust value from the piece of evidence (for simplicity called T_{evd}), be that from an observation (after the application of the *eval()* function) or from a recommendation (note that the recommender is just providing a trust value).

τ-function. This function takes the current trust value (T_{curr}) and the trust value from evidence (T_{evd}) and produces a trust-attraction value (τ-*value*). Formally we can describe the function as:

$$\tau(T_{curr}, T_{evd}) = \tau\text{-value}$$

These trust-attraction values belong to a domain that is divided into trust increasing, trust decreasing and trust neutral values. A trust increasing value cannot lower, according to the trust ordering, our current trust value. For example, if T_{curr} is *[medium, medium]* and T_{evd} is *[medium,high]* that means the new trust values cannot be lower than *[medium, medium]*. Similarly a trust decreasing value cannot increase our current trust value. While a trust neutral value can neither increase nor decrease our current trust value. If the trust-attraction value is positive, negative or neutral depends on the relative position on the trust ordering lattice of the two trust values, arguments of the τ-function.

ι-function. This function also takes the current trust value T_{curr} and the trust value from evidence (T_{evd}) and produces an information-attraction value (ι-*value*). Formally we can describe the function as:

$$\iota(T_{curr},T_{evd})= \iota\text{-value}$$

These information-attraction values belong to a domain that is divided into conflicting, reinforcing and neutral values. To determine which value of the domain the ι-*value* belongs to, we follow these conditions:

- ▷ The maximum reinforcing value is returned when the greatest lower bound (g.l.b.) according to the information ordering of the two arguments is equal to the current trust value(T_{curr}).
- ▷ The maximum conflicting value is returned when the g.l.b. of the two arguments is equal to the least element of the information ordering.
- ▷ A neutral value is returned when the g.l.b. of the two arguments is equal to the trust value from the evidence.

The exact measure of the ι-attraction value has the following relationship to the strength (number of labels included in interval) of the arguments and the overlap(number of common labels):

(Strength (evidence trust)/Strength (current trust))*(Overlap)

The maximum strength for a trust value is assigned to values that have no other trust value greater than them according to the information order, while the minimum strength is assigned to the least element of the information ordering c.p.o. The rest of the strength values reflect the number of trust values that are greater than the value in question according to the information ordering. The overlap reflects the level of contradiction or reinforcement of the two trust values and as was described above depends on the relative position of our current trust value to the g.l.b. of two arguments of the ι-function.

Applying the two attraction functions (ι-function and τ-function) to the new piece of evidence produces a pair (τ,ι)that will be used to update the current trust value (T_{curr}).

The update function. The update function, *update()*, takes as input the pair (τ, ι) and the current trust value (T_{curr}), producing a new trust value (T_{new}).Formally, we can describe the *update()* function as:

$$\text{update}((\tau, \iota),T_{curr})=T_{new}$$

The two attraction values are in fact excluding certain trust values as a possibility for the new trust value. The τ-attraction excludes values based on their position on the trust ordering lattice, while the ι-attraction excludes values based on their position on the information ordering c.p.o. For example, a trust increasing value τ-value would exclude all trust values for which the current trust value is greater than according to the trust ordering. Similarly, a contradicting value ι-value will exclude all trust values that are greater than the current trust value according to the information ordering. These

functions will result in two subsets (τ-set and ι-set $\subseteq T$) of trust values each consistent to the evaluation of the new piece of evidence according to the each ordering. The new trust value $T_{new} \in \tau$-set $\bigcap$ ι-set. The exact functions for the determination of the new trust value depends on the application.

2.4 Evidence Store

In our model the position is taken that the trust information is stored in order to allow us to reason about principals' trustworthiness in future interactions. Therefore, we introduce a structure called the *evidence store*. The use of a layered structure (Figure 2) to store trust information provides a greater depth of information upon which to base any decision than merely storing the individual trust values relating to the entity in question.

The following points discuss these layers of the evidence store and the operation functions between the layers:

1. The base layer contains lists of all observations or recommendations. The output of the evaluation function `eval()` will be stored in these lists. There is a fixed size for each one of these lists. When adding a new experience, the size of the list (L) will be checked. If L has some space, then the new piece of evidence will be added to the previous ones. When L is full, the attraction of old observations or recommendations will be calculated and then cleared so as to add the new one. In this way, all evidence in the history of the principal's interactions is reflected in the trust value even though the observations themselves are discarded. The trust update function *update()* produces a trust value (T_{obs})that will be stored on the second layer. The update of the evidence will take place separably, for recommendations and observations, to allow us to determine a trust value that purely based on observations (T_{obs}). This facilitates the process of giving recommendations upon receiving recommendation requests.
2. The second layer in the structure contains a trust value (T_{obs}) specific to observation and a trust value (T_{rec}) reflects the attraction of all the received recommendations. The new final T_{curr} to be used for decision making can be calculated using the attraction notion in two ways. The first approach is to apply the attraction of T_{rec} on T_{obs} to produce a new trust value that will be stored as (T_{curr})in the top layer. In the second approach we apply the attraction of both T_{rec} and T_{obs} on the current trust value (T_{curr})that is stored in the top layer and produce a new trust value. The result of these two operations produce a trust value that will be stored in the top layer.
3. The value in the top layer influences the trust engine, which invokes the trust method. In this way, the local trust policy will be updated in the light of this new trust value.

3 Trust Policy languages

In this section we show how to express what was said in the previous sections using a language for describing policies.

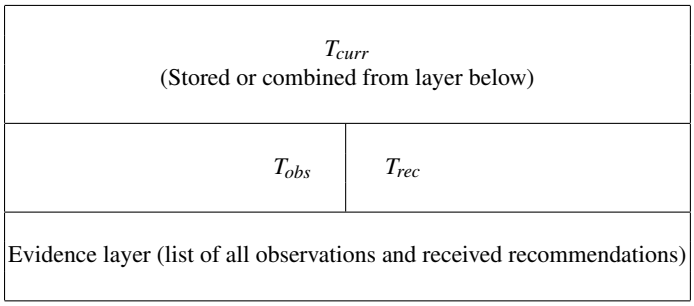

Fig. 2. The Trust Information Structure

3.1 The Policy Language

In [2] we defined a general setting, where policies are specified via a language. Formally, policies are functions of type $\mathcal{P} \to \mathcal{T}$ such that for each principal a they return a trust value. The language permits the specification of these functions. A policy language has the form $\lambda x : \mathcal{P}.\tau$ where τ specifies the returned value of the policy, done by using certain operators. One of these operators is the operator $\ulcorner \cdot \urcorner$, called *delegation*. For instance the policy $\lambda x : \mathcal{P}.\ulcorner a \urcorner(x)$ specifies the function that, for any principal $b \in \mathcal{P}$, returns a's trust in b. Delegation makes policies dependent on the global trust m. The function m is defined as the least fixed point of all policies (see the reference for more details and formal semantics).

The operator of delegation can also be combined with other operators provided from the language. For instance we could write a policy which says that our trust in b is a's trust in b but for any other principal c it will be the least upper bound between a's trust in c and a threshold value $[d_0, d_1]$:

$$\lambda x : \mathcal{P}.(x = b) \mapsto \ulcorner a \urcorner(x); ([d_0, d_1] \sqcap \ulcorner a \urcorner(x))$$

where the operator $\cdot \mapsto \cdot; \cdot$ is semantically equivalent to an if-then-else and $\sqcap$ is the l.u.b. in lattices.

3.2 Encoding in the policy language

The encoding in the policy language is done by exploiting two features: the syntax of the policy and the structure of the set $\mathcal{T}$. When we give a policy π written in the language we give an algorithm for computing a function. Hence, when writing the policy, we "store" some information in the syntax of the language which means that it is possible to record the current trust value.

Representing the various layers of the evidence repository could be handled by giving more structure to the set $\mathcal{T}$. Suppose that the initial set of trust values (as the one of the previous sections) is $\mathcal{T}_{val}$, provided with an information ordering $\sqsubseteq$ and a trust ordering $\preceq$ as required. Then we can define a new set $\mathcal{T}$ as

$$\mathcal{T} = [EVD] \times \mathcal{T}_{obs} \times \mathcal{T}_{rec}$$

where $[EVD]$ is the list of observations, *e.g.* $[evd_1,\ldots,evd_n] \in [EVD]$ for $evd_1,\ldots,evd_n$ evidences. This means that our policy contains a list of evidences (as shown in the base layer of the evidence repository), and two trust values (as shown in the second layer): the first one which refers to the current trust value based on observation (which is then a constant in the syntax), and the second one which refers to the value updated with recommendations.To describe these trust values as a policy, we will initially start from the situation where there is no observations or recommendations. Therefore, the policy could be defined as:

$$\pi_a = \lambda x : \mathcal{P}.([],\bot,\bot)$$

where $[]$ is the empty list and $\bot$ is the bottom of the information ordering (no information).

When we have an observation, say o_1, we just need to update the list of observations and so get the the new policy

$$\pi_a = \lambda x : \mathcal{P}.([evd_1],\bot,\bot)$$

In a situation when the returned trust value that we asked the trust engine for is not enough, we might ask principal b for recommendation about c. Then our policy would be updated to

$$\pi_a = \lambda x : \mathcal{P}.([],[d_0,d_1],(x=c) \mapsto update((\tau_1,\iota_1),[d_0,d_1])$$

where $\tau_1 = \tau((\ulcorner b \urcorner(x).2),\ulcorner a \urcorner(c).2))$ and $\iota_1 = \iota((\ulcorner b \urcorner(x).2),\ulcorner a \urcorner(c).2))$ determines the attraction. The operator .2 (in general .*n* for any natural number) stands for the projection of the 2-nd element (in general *n*-th) over the tuple (over set $\mathcal{T}$). Hence every time we update we keep the old value and add new references to other principals.

Notice that when getting an observation we just store it in the list but we do not update the trust value. That is going to be done with a further update which basically cleans the list and updates (with the function `eval`) the trust value in the middle of the triple.

4 Conclusions and Further Work

The primary contribution of this paper is in formalizing a model for a trust-based decision maker that incorporates four components: trust engine, risk evaluator, trust lifecycle, and evidence repository. The model reflects on how important is to take the two factors of trust in risk on board in establishing collaboration between principals. The model showed that reaching decision is a prolonged process to remove some shade of uncertainty issues concerning risk and principals' trustworthiness.

A similar approach for trust management formalization is described in [3]. This model also accommodates and correlates trust and risk in establishing interactions in the context of e-services. In their proposed model, there is no formal definition of trust. Alternatively, they described a classification of trust relationships in the environment of e-commerce. It is also important to mention that subjective logic theory [8] , has influenced our addressing of issues of uncertainty in our model.

There are a number of promising directions for further investigation. One of these issues is how to introduce complexity in the model by considering more complex collaborations (i.e. a collaboration that takes place between more than two collaborators to perform dependent actions). An interesting area of research is how to make a decision based on the relative context and time. This implies encoding these two notions in the model. For the trust engine, one particular area of interest is to complement the denotation model presented here with an operational model where, for instance, we will need to address the question of computing trust information efficiently over the global network.

Our future research will be focused on examining this model by developing a simulation framework. In this framework, we will study the properties of the proposed model. The results of the simulation and the study of the current model properties will guide us to more complex collaboration model. We hope to report on this work shortly.

Acknowledgements The work in this paper is supported by SECURE (Secure Environments for Collaboration among Ubiquitous Roaming Entities, IST-2001-32486) funded by the EU FET Programme under the Global Computing Initiative.

References

1. Jean Bacon, Nathan Dimmock, David Ingram, Ken Moody, Brian Shand, and Andy Twigg. Definition of risk model. *SECURE Deliverable 3.1*, 2002.
2. Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A formal model for trust in dynamic networks. In *Proc. of International Conference on Software Engineering and Formal Methods*, September 2003.
3. Theo Dimitrakos. System models, e-risks and e-trust.towards bridging the gap? In *Proceedings of the 1st Conference on e-Commerce, e-Business, e-Government.Kluwer Academic Publishers*, October 2001.
4. C. English, P. Nixon, S. Terzis, A. McGettrick, and H. Lowe. Security models for trusting network appliances. In *Proc. of the 5th IEEE International Workshop on Networked Appliances*, October 2002.
5. C. English, W. Wagealla, P. Nixon, S. Terzis, A. McGettrick, and H.Lowe. Trusting collaboration in global computing. In *Proc. of the First International Conference on trust management*, May 2003.
6. Colin English, Sotirios Terzis, Waleed Wagealla, Paddy Nixon, Helen Love, and Andrew McGettrick. Trust dynamics for collaborative global computing. In *Proc. of the WETICE conference*, 2003.
7. Tyrone Grandison and Morris Sloman. A survey of trust in internet application. *IEEE Communications Surveys, Fourth Quarter*, 2000.
8. Adun Jøsang. A logic for uncertain probabilities. *Fuzziness and Knowledge-Based Systems*, 9(3), 2001.
9. SECURE Project. Official website. `http://secure.dsg.cs.tcd.ie`, 2002.