MATE: Mobility and Adaptation with Trust and Expected-utility

Daniele Quercia*
Department of Computer Science,
University College London, London, WC1E 6BT, UK
E-mail: d.quercia@cs.ucl.ac.uk
*Corresponding author

Stephen Hailes
Department of Computer Science,
University College London, London, WC1E 6BT, UK
E-mail: s.hailes@cs.ucl.ac.uk

Abstract: Mobility is one of the defining characteristics of many pervasive environments. As pervasive devices move, they experience new contexts (e.g., new network boundaries or new peering devices) to which they may adapt by loading new software components or by updating existing ones. Unfortunately, loading and updating code introduces security problems, particularly because components may be from untrusted sources with which devices happen to be networked at the time of need. Thus, a pervasive device must be able to distinguish between trustworthy and untrustworthy peering devices so that they load software components only from trustworthy ones. To achieve this, the device may run a software agent that comprises a trust model and a decision-making model: the former manages information about peering devices' trustworthiness upon which the latter decides appropriate actions, e.g., whether to load software or ask the user for further guidance. We here propose a decision-making model named MATE, which works as follows. It lists potential risks depending on the device's context and running application. When it has to decide whether to load software from another device, MATE assigns occurrence probabilities to those risks in two ways. In the first approach, the probabilities depend on the peering device's trustworthiness (i.e., the output of the trust model). In the second approach, MATE computes the probabilities using a Bayesian formalization that works without an additional (trust) model. After computing those probabilities, MATE applies the expected-utility decision rule to decide whether or not it is in its node interests to load a particular component.

Keywords: risk; expected utility; trust; reputation; mobility; mobile code; decision-making model; component model.

Reference to this paper should be made as follows: D. Quercia and Stephen Hailes (2006) 'MATE: Mobility and Adaptation with Trust and Expected-utility', *Int. J. Internet Technology and Secured Transactions (IJITST).*

Biographical notes: Daniele Quercia is a PhD student in the Computer Science department of the University College London, working in the Networking Research Group. His research focuses on enabling mobile resource sharing in the presence of untrustworthy devices. Before commencing a PhD in the department, he was a Research Fellow on the SEINIT European project. He studied at Politecnico di Torino (PoliTO), Italy, for his Master of Science in Computer Engineering. As a recipient of international scholarship awards (while at PoliTO), he was a visiting student at Universitaet Karlsruhe, Germany, and University of Illinois, Chicago, U.S.

Stephen Hailes was an undergraduate at Trinity College Cambridge and then a PhD student in the Cambridge University Computer Lab. Following this, he joined the Department of Computer Science at University College, London, as a Research Fellow. Shortly afterwards he was made a lecturer, and then, in 2000, a senior lecturer. He has been Director of Studies since January 2003. He is interested in all aspects of mobile systems and security. But most particularly in ad hoc systems, pervasive/ambient computing environments, and how to secure these areas. He is currently the PI of the MARS project, which is a collaborative project in conjunction with BT looking at building middleware components for trust management in pervasive environments, and the CoI on SENIT, an EC Framework 6

Integrated Project looking at securing ambient networks. His past projects include 6WINIT, an EC Framework 5 project, EPSRC-funded projects (PIMMS, MOSQUITO, and MEDAL), and UKERNA-funded projects such as ACOL and TITAN.

1 INTRODUCTION

The term pervasive (or ubiquitous) computing describes a computational environment in which devices that are mobile and sometimes resource constrained (e.g., sensors and embedded systems) interact to obtain and provide services anywhere and anytime. Application developers and vendors will not be able to pre-load all the required software components on those devices, both because they are, by their nature, resource constrained (e.g., sensors) and because there is a need to patch and update software from time-to-time (e.g., PDAs). Hence, those devices will have to rely on loading the components dynamically and as necessary from other devices.

Installation of software components requires additional security mechanisms or assurance that the components work as they claim to be - for example, that they do not contain trojans. In traditional systems, digitally signing software supports assurance by assuming the existence of globally trustworthy CAs whose public keys are built into browsers and operating systems. Whilst the number of software providers remains limited, this is a reasonable solution; however, even in the wired Internet, small software providers often distribute code using shareware and freeware mechanisms, with assurances that are at best limited and built around unpoliced reputation rating systems on well known shareware sites. In pervasive environments, scale and dynamicity increase, and full network connectivity to the wider Internet cannot always be assumed; on the other hand, the need to load new software components for appropriate to prevailing conditions may be urgent if the device is to continue to operate as the user expects.

Consequently, for the purpose of loading software components, mobile devices cannot assume that they will always be able to obtain software from trusted sources; on the contrary, to provide the necessary degree of availability, they may need to obtain software from other devices in a peer-to-peer fashion. This clearly represents a potential danger and there is a need for a decision procedure balancing the risks of loading such components against the potential utility that they would provide. This argument is little different from that employed by those researching trust in pervasive systems: that the addition of local decision making mechanisms allows an appropriate balance to be made between uncertainty and utility. the necessary degree of availability, they may need to obtain software from other devices in a peer-to-peer fashion. This clearly represents a potential danger and there is a need for a decision procedure balancing the risks of loading such components against the potential utility that they would provide. This argument is little different from that employed by those researching trust in pervasive systems: that the addition of local decision making mechanisms allows an appropriate balance to be made between uncertainty and utility.

Software agents that engineer social (human) trust may represent security mechanisms appropriate for ubiquitous devices. An agent, acting on behalf of a device's user, exchanges software components only with other *trusted* devices, whilst minimizing both user intervention and resource consumption. The agent considers trusted those devices that, for example, it has found to be reliable in past software exchanges - they have always cooperated supplying what they were expected to.

Furthermore, feeding trust-related information into decision-making processes may lead to improved security mechanisms. Those processes carry out trust-informed reasoning while being aware of potential risks. This is desirable because how much trust is necessary for exchanging software components strictly depends on the level of risk involved. Risk and trust are in an inverse relationship (Patrick, 2002): the riskier an activity is, the higher is the trust level required to engage in such an activity. For instance, the level of trust required for devices that provide components may likely be higher when loading security or core middleware components, than when loading a new electronic game that does not implement critical security features, has been downloaded just-for-fun, and can run within a sandbox.

The body of work in integrated management of trust and risk is rather general and, as a consequence, does not directly apply to issues related to mobile software components. In the few cases of work on specific scenarios, great emphasis has been given to issues of access control in pervasive computing.

We present a novel decision-making model, named MATE, that integrates both risk and trust reasoning. More specifically, the model is part of a software agent running on a pervasive device and reasons about loading software components from other devices. The expected utility theory forms the basis of such reasoning, which accounts for trust information about software sources, risk attitudes of the device user, and context information.

In the remainder of the paper, Section 2 discusses related work. In section 3, we introduce a scenario intended to be illustrative of a real-life usage of the model. Section 4 states the assumptions we make in modelling MATE. In section 5, we describe all elements of MATE. Section 6 illustrates the experiment we carried out to evaluate MATE. In section 7, we make a clear distinction between trust and assurance and

we then discuss in depth our solution. Finally, section 8 describes future work and section 9 presents our conclusion.

2 RELATED WORK

Researchers have long realized the need to deal with trust and reputation in pervasive computing and have proposed many approaches. The notion of trust management has formed a subject of study since work by Blaze, Feigenbaum, and Lacy (1996) with their seminal paper on decentralised trust management, which shows a language for specifying trusted actions and trust relationships; they also describe a prototype implementation of a trust management system, called PolicyMaker. Blaze *et al.* (1998) then introduced KeyNote which is a credential-based distributed policy management framework: it puts specific emphasis on access control decisions rather than general trust management (e.g., it does not address trust evolution issues). Abdul-Rahman and Hailes (2000) then proposed a distributed trust management model, with which entities could autonomously reason about trust, without relying on a central authority. Mui (2002) then investigated a computational model of trust, in which the concept of reputation was introduced. More recently, Capra (2004) proposed a human trust management model, specifically for mobile systems and applications. Quercia *et al.* (2005) proposed STRUDEL, a Bayesian trust management framework for bandwidth sharing in mesh networks. They then put forward B-trust, a generalization of such a framework (Quercia *et al.* (2006)).

As distributed trust management frameworks started to have a profound impact on the computing research scene, consideration started to be given to risk analysis aspects which relate to trust (Chen *et al.*, 2003). The SECURE project (Cahil *et al.*, 2003) incorporated a formal trust model (Carbone *et al.*, 2003) and an explicit model of risk (Dimmock, 2003) for assessing collaborations in ubiquitous computing environments. The trust management model dynamically builds trust from local trust policies, based on past observations and recommendations. Within the same project, Wagealla *et al.* (2003) presented a formal model for the management of trust life-cycle issues, with consideration of both trust and risk. Although foundational, the SECURE risk model (Dimmock, 2003) has a limitation: it computes utility values on probabilities of events, whereas risk theory in economics suggests that utility values should be not estimated on probabilities of events, but rather on events (i.e., the utility of the *probability* of raining cannot be defined, as opposite to the utility of raining).

More recently, trust-based risk investigation has led to the integration of the expected utility theory in computational risk models. Josang and Lo Presti (2004) analyzed the relationship between risk and trust and derived a computational model integrating the two concepts; however, their model focuses on using risk to deduce trust, whereas we use the opposite approach: we view trust as driving risk (English *et al.*, 2004) and, thus, integrate the trust assessment within a global risk-aware decision framework. Dimmock *et al.* (2005) investigated a model of risk again based on the expected utility theory and evaluated it in a scenario of P2P collaborative spam detection. This work, however, focuses on issues of trust-based access control and, consequently, does not tackle security risks associated with mobile software components.

This paper extends our previous effort (Quercia *et al.*, 2005) in that we extend MATE by designing and evaluating a novel Bayesian revision approach.

3 SCENARIO

The following scenario will help our exposition. Whilst in her office, Alice initiates a video-conference with Bob using her PDA, which has limited computational power and constrained storage space. Now Alice realizes she has to go home, but, at the same time, she does not want to stop the conversation with Bob. She thus decides to continue conferencing on the move. The architecture running on Alice's PDA should guarantee secure communication across all traversed physical spaces, whilst the video-conference is seamlessly carried out. Alice often uses the video-conferencing tool in the office, but very rarely outside. Consequently, her PDA keeps the video-conference software permanently stored but only loads the security components that support secure outdoor communication when necessary. Thus, the PDA's architecture loads appropriate software components as context changes:

- **At the office** - Alice's PDA has the videoconferencing tool stored locally. No security component is needed as the office is considered physically secure and *collaborative*.
- **From office to home** - Alice is now traversing spaces which are considered *non-collaborative* (e.g., public spaces such as streets and passenger coaches). Alice's PDA therefore needs to discover and load two additional software components: a tool for wireless network access control (e.g., PANA (Forsberg *et al.*, 2003)) and a software component for confidentiality protection (e.g., IPSec (Kent and Atkinson, 1998)).
- **At home** - Alice's PDA keeps the component for confidentiality protection loaded but marks as removable the code for wireless network access control because it is rarely used and a base station now provides network access in Alice's home[1]. The PDA still needs to keep the communication to the base station confidential, since the traffic can easily be overheard or spoofed by Alice's neighbors and passers-by.

[1] Viewing the PDA as a cache, there is no real need actively to remove this component, but, to aid the cache replacement algorithm, it should be marked as being of lower priority than when it was in active use.

The actors in the scenario fall into three classes: Alice's PDA is a *component loader* as it discovers and loads software components, normally those that are rarely used; *component suppliers* are the devices that provide the software components; and *component authors* are the organizations or individuals that authored the components. Thus, when the component loader needs a software component (e.g., IPSec), it attempts to discover nearby devices (component suppliers) willing to provide a component with the requisite semantics and within the time frame necessary.

Component suppliers reply with details of appropriate components, and a service level that denotes how confident they are in being able to deliver the component within the time frame requested. The component loader uses MATE to select which of the component suppliers to use, based on the probabilities of potential risks, which in turn depend on (i) the trust level in the component author and guarantees about the integrity of the component, (ii) the claimed service level, and (iii) the trust in the supplier to provide the components as detailed. In fact, if a component loader decides to accept a piece of code, it implicitly accepts some risks.

For example, the chosen component supplier may be malicious and may consequently deliver code out-of-time or may ship malicious software (e.g., a component may carry a virus). To mitigate potential risks, the component loader takes informed decisions based on component suppliers' trustworthiness: it decides whether the component supplier is believed to generate more benefits (e.g., the component that does what it is supposed to) than risks (e.g., it is an unreliable component).

4 ASSUMPTIONS

Before describing MATE, let us introduce the *assumptions* behind its design. First, we assume applications are composed of locally interconnected components. There is considerable research within the mobile middleware community to support such a model; for example, Beanome (Cervantes and Hall, 2000), Gravity (Cervantes and Hall, 2004), SATIN (Zachariadis *et al.*}, 2004) and OpenCOM (Clarke *et al.*, 2001).

Second, we assume that components are available with their dependencies.

Third, we do not deal with the aspect of component discovery. For this purpose, any appropriate discovery framework available in the literature may help; for example, RUBI (Harbird *et al.*, 2004) is a resource discovery framework for mobile devices in which discovery is based on devices' local views of the network structure.

Finally, we restrict our scenario to risks that take place if component suppliers do not provide software within the agreed time ranges. In other words, we do not examine the potential risks that may occur from the way a software component actually operates once delivered, although such considerations are a logical extension of this simple example. This situation might arise if the software component is signed with the author's key, thus guaranteeing the software integrity, and the component-loader highly trusts the author: in this case, the component-supplier is entirely irrelevant to the maliciousness or not of the software and the component-loader only cares that the software will be supplied on time.

The scenario resulting from these assumptions is the following. Alice's PDA needs a software component. Bob's PDA is willing to provide it and declares a *promised delay* d_p (e.g., of 10 seconds) with a confidence *CL* (e.g., of 80%). We denote by d_e the *experienced delay*, i.e., the delay Alice's PDA will actually experience.

5 THE DECISION-MAKING MODEL: MATE

In economics, the expected utility (EU) theory often models decisions under uncertainty. Bernoulli (1954) ¯rst suggested that people maximize not the expected monetary value of an outcome, but its expected utility. Formally,

$$EU = \sum_i \pi_i \cdot u(o_i),$$

where *EU* is the expected utility, π_i is the probability that the i^{th} outcome will occur and $u(o_i)$ is the utility of the i^{th} outcome. Von Neumann and Mongestern (1964) then showed that the maximization of expected utility models a rational behaviour, given the validity of 6 axioms. In particular, their formulation assumes that *objective* outcome probabilities can be always defined. Amongst others, Savage (1954) developed a version of EU theory in which probabilities are subjective.

MATE is based on EU theory and has the following properties: (i) outcome probabilities are not objective, but rather subjective, as they depend, as we will see, on subjective evaluations, such as trust values; (ii) we define utility functions not through conventional economics methods (e.g., lottery technique (Hirshleifer and Riley, 1992)), but through a simple, yet novel policy refinement process, which accounts for risk attitudes of the device user.

The remainder of this section elucidates our model's building blocks. First, we introduce all the elements around which we model MATE. We will then dwell on two elements: elementary utility function and set of state probabilities.

5.1 Elements of MATE

The component loader (Alice's PDA) bases its decision whether to accept a component from any component

supplier, e.g., Bob's PDA, on the following elements (Hirshleifer and Riley, 1992):

1. A set of *actions* (a_1;…; a_x; … ; a_A) that Alice's PDA can carry out. For example, possible actions include '*accept component*', '*do not accept component*' and '*ask the user for further guidance*', as listed in the first column of the table in Figure 1.

2. A set of *states* (s_1; …; s_y; …; s_S), i.e., the set of occurrences represented within the model. For example, states might include '*component supplier delivers within an acceptable time range*', '*component supplier delivers a malicious component*', '*component supplier delivers a component from a highly trustworthy author*' and '*component supplier does not deliver any component*'. More specifically, a state could be that the actual delay d_e Alice's PDA experiences in receiving the component from Bob's falls in a range that allows Alice's video-conference to continue seamlessly. For the purpose of simplicity, we define three possible ranges of delay d_p: the seamless range (R_1), the limited disruptions range (R_2) and the risky range (R_3) (see the first row of the table in Figure 1).

3. A set of *outcomes*. We denote with $o(a_x; s_y)$ an outcome function that returns the outcome corresponding to the case in which Alice's PDA carries out the action a_x while the state s_y takes place. For example, the combination of the action '*no component accepted*' and any possible states would result in the outcome '*give up the current activity*', if the component in question is absolutely necessary and confidentiality is a prime requirement (e.g., a piece of code that performs a robust encryption algorithm supports the confidentiality of the video-conference; if such a piece of software is neither locally stored nor remotely available, the video-conference must be stopped). The table in Figure 1 is the outcome matrix and shows the outcomes corresponding to all combinations of action and state.

4. An *elementary-utility* function $u(o_z)$ that measures Alice's desirability of the any given outcome o_z. Subsection 5.2 discusses in depth the function and the factors affecting it.

5. A set of *state probabilities*. Each state probability $\pi^t_{(Bob's\ PDA)}(s_y)$ expresses Alice PDA's belief that state s_y will take place if interacting with Bob's at time t. For example, if Alice's PDA believes that Bob's is very trustworthy in delivering software, it may assign a very low probability to the occurrence of the state '*component supplier delivers a malicious component*', whereas it believes the state '*component supplier successfully delivers the component*' more probable. Similar considerations may be extended to the trust level of the component author: if the integrity of the software component is guaranteed and Alice's PDA highly trusts the component author, then the probability of the state '*component supplier delivers a malicious component*' is close to the minimum value, for example. Subsection 5.3 shows the computation of the state probabilities.

6. The expected-utility *decision rule* which Alice's PDA applies so to decide the most advantageous action. Given a set of possible actions such as '*carry on the videconference*' or '*give up the video-conference*', Alice's PDA derives an ordering of such actions, by computing the utility for each of them. It does so for each device willing to be a component supplier. If we consider a given component supplier *h*, the component loader's utility for the action *a* at time *t* is

$$U^t_{a,h} = \sum_{i \in [1,S]} \pi^t_h(s_i | d_p \in R_\rho) \cdot u^t(o(a, s_i)),$$

where $\pi^t_h(s_i \mid d_p \in R_\rho)$ is the probability that state s_i takes place at time *t*, given that *h* promises a delay d_p within range R_ρ; if s_i takes place and Alice's PDA carries out the action a, the utility of the resulting outcome is denoted by $u^t(o(a, s_i))$. For a given action, the component loader maximize $U^t_{a,h}$'s over all the devices willing to be component suppliers. Thus, it obtains the preferred supplier for each possible action. It then chooses the action with the highest utility value.

So, to apply the expected-utility decision rule, we need to define both the elementary utility function and the computation of the state probability. Both are the subjects of the next subsections.

ACTIONS	STATES		
	CS **delivers** *C* **within** R_1	*CS* **delivers** *C* **within** R_2	*CS* **delivers** *C* **within** R_3
Accept component	Carry on seamlessly	Carry on with limited disruption	Give up
Do not accept component	Give up	Give up	Give up
Ask User	Alice interacts with GUI	Alice interacts with GUI	Alice interacts with GUI

Figure 1 *Matrix of outcomes of three alternative actions and three possible states: the component supplier (CS) delivers the component C with delay d_e, which can be in the seamless range R_1 (first state), or in the limited disruptions range R_2 (second state), or in the risky range R_3 (third state).*

5.2 Elementary utility function

In general, elementary utility functions describe both absolute and relative preferences of users of experiencing a given outcome. *Preference elicitation* is the process of extracting preference (utility) information from a user and still represents an open research issue for one main reason: people find it difficult to attach utilities to outcomes. For instance, one may prefer one pub over another, but how much a particular pub is preferred is often difficult to express.

For the purpose of this paper, a simple policy refinement process elicits user preferences: from high-level user policy specifications the component loader device (Alice's PDA) extracts the appropriate parameters (denoted by w_i) to attach an elementary utility ($u(o)$) to each single outcome o (the table in Figure 1 shows a list of possible outcomes). In so doing, we consider the risk attitude of Alice.

More precisely, MATE computes the elementary utility $u(o)$ of a single outcome o when its hosting device (Alice's PDA) runs a video-conferencing application in two steps. First, it lists *preferable occurrences* that positively contribute to Alice's utility when running the video-conference. For example, those might include *absence of disruptions* (*AD*), *spared user time* (*SUT*), and *inverse security gap* (*ISG*), all of which we will discuss later in this section.

Second, MATE sums all single *positive contributions* $PC_i(o)$ ($\in$ [0,1]) of the outcome o to each i^{th} preferable occurrence. It then weights this sum to obtain the elementary utility $u(o)$. Examples of positive contribution (*PC*) are:

1. *PC* to *absence of disruptions* ($PC_{AD}(o)$) - For outcomes allowing the video-conference to be carried on seamlessly, this *PC* approaches the maximum of 1, whereas for outcomes leading to permanent interruption, it falls to minimum 0. A *PC* of the outcome '*carry on with limited disruptions*' would assume an intermediate value closer to the maximum than the minimum - say a value of around 0.8.

2. *PC* to *spared user time* (PC_{SUT} (o)) - Outcomes causing massive Alice's intervention bring this *PC* down to the minimum 0, whereas the maximum value 1 is only experienced when Alice is not interrupted.

3. *PC* to *inverse security gap* ($PC_{ISG}(o)$) - To clarify the meaning of this *PC*, let us imagine the following situation. Alice's PDA wishes to load a component to reach a desired security level, denoted by Sec_A. After loading a component, it reaches security level Sec_B. The value of this *PC* is close to maximum value 1, if security gap between the needed level of security and the one that is effectively reached has been completely filled (best case). Otherwise, it decreases as the security gap increases. More formally, this positive contribution can be expressed as

$$PC_{ISG}(o) = \begin{cases} \frac{1}{Sec_A - Sec_B}, & \text{if } Sec_A - Sec_B > 0; \\ 1, & \text{otherwise.} \end{cases}$$

This PC refers to general security; however, it would be more appropriate to have one positive contribution to each specific security aspect, such as confidentiality, integrity and authentication. We weight each $PC_i(o)$ with a factor w_i expressing how much Alice cares about the i^{th} preferable occurrence. These factors reflect Alice preferences, which her PDA stores as high-level policy and upon which then performs a policy refinement process: it extracts from high-level policy specification each w_i factor. For example, Alice specifies through a GUI how much she cares about disruptions upon which her PDA then extracts a value for w_{AD} (i.e., her preference in having absence of disruptions (*AD*)).

To sum up, the elementary utility of the outcome o, which falls in the range [0,1], is

$$u(o) = \frac{\sum_{i \in [1,n]} w_i \cdot PC_i(o)}{\sum_{i \in [1,n]} w_i},$$

where w_i expresses how much Alice cares about the i^{th} preferable occurrence, and $PC_i(o)$ is the positive contribution of the outcome o to the i^{th} preferable occurrence.

5.3 Set of state probabilities

A state probability expresses the belief of Alice's PDA that a certain state will take place when interacting with Bob's (denoted by h).

More specifically, given a state '$d_e \in R_1$' (Alice experiences a delivering delay within range R_1 - this is one of the states listed in the first row of the table in Figure 1), we have two approaches to compute the probability of its occurrence. We first describe an approach that considers MATE integrated with a trust model. We then describe the second approach that does not depend on any additional (trust) model and is based on a Bayesian formalization.

A MATE for a trust model

Let *f(x)* be the density function for the delay d_e (assumed to be random). The probability that d_e falls in, say, the seamless range R_1 is the integral of *f* on that range; in other words:

$$\pi_h^t(d_e \in R_1) = \int_{R_1} f(x)dx. \quad (1)$$

We can similarly compute the probabilities for the remaining ranges. For the purpose of this paper, we suppose that *X* is a normal distribution. We intend the choice of the normal distribution to be illustrative rather than prescriptive and, as a consequence, the decision scheme can support other types of probability distribution, which will, naturally, affect the calculations below. As we will see, the computation of the probability of each state does not require the computation of the integral above, but simple table lookups will do.

In general, two terms describe the normal distribution: its mean μ and standard deviation σ. *X*'s mean equals d_p. Having defined μ, to calculate expression (1), we need *X*'s standard deviation, which represents how fast *X* decreases as we move away from the mean. The degree of deviation from the mean (i.e., variance of *X*) decreases, as the uncertainty diminishes; this happens when the following factors increases: (i) trust level *T* in Bob's PDA to deliver on time; (ii) confidence level *CL* on the declared delay d_p. As such, we can state that:

$$P(d_e \in U^+(\mu)) = \alpha T + \beta CL, \quad (2)$$

where α and β are positive constants, such that $\alpha+\beta = 1$. In other words, the probability that the experienced delay falls in a range close to the mean, denoted by the right neighbourhood $U^+(\mu)$ of mean μ, solely depends on both trust level *T* and confidence level *CL*. To define the right neighbourhood of μ, we choose a small positive real number δ so that $U^+(\mu) = \{x \in R \mid 0 < x-\mu < \delta\}$. We include only the right neighbourhood because we consider only the events in which the delivery time is worse than the declared one.

As the quantity $\alpha T+\beta CL$ increases, the concentration of values of *X* that fall within right neighbourhood $U^+(\mu)$ increases. This is because the uncertainty (i.e., *X*'s variance) decreases and consequently more values populate the area close to the mean.

Since *X* is a normal distribution with mean μ and variance σ^2, we can define $Z = (X-\mu)/\sigma$, which is the standard normal distribution, that is, a normal distribution with mean equal to 0 and unitary standard deviation. We can then write:

$$\begin{aligned} P(d_e \in U^+(\mu)) &= P(\mu < X < \mu + \delta) = \\ &= P(0 < Z < \frac{\delta}{\sigma}) = \\ &= \phi_Z(\frac{\delta}{\sigma}) - \phi_Z(0) = \\ &= \phi_Z(\frac{\delta}{\sigma}) - \frac{1}{2}, \end{aligned} \quad (3)$$

where ϕ_Z is the cumulative distribution function of standard normal distribution *Z*.

We finally combine equation (2) and equation (3) to obtain the functional dependency of *X*'s standard deviation σ on the trust level and on the confidence level:

$$\sigma = \frac{\delta}{\phi_Z^{-1}(\alpha T + \beta CL + \frac{1}{2})}. \quad (4)$$

Given trust level *T* and confidence level *CL*, we can compute the probability of each state. Equation (4) gives standard deviation of *X*, which has a known mean: it equals d_p, the delay Bob's PDA promised. Having X's mean and standard deviation, we can easily evaluate equation (3), which computes the probability of a given state. Having μ and σ, we can convert *X* into the standard normal distribution *Z*, which has been extensively tabulated thus allowing us to compute the integral (1) with simple table lookups.

After receiving the software component, Alice's PDA knows the actual delivering delay d_e upon which its trust model updates the trust in Bob's: the lower the difference between d_e and d_p (delay Bob's PDA promised), the better the update.

An independent MATE: a Bayesian revision approach

Let us now see how MATE running on Alice's PDA computes a state probability $\pi^t_h(d_e \in R_\gamma)$ through a Bayesian revision without any additional (trust) model. More precisely, we are interested in $\pi^t_h(d_e \in R_\gamma \mid d_p \in R_\rho)$ - the probability of Alice's PDA experiencing a delay d_e within range R_γ when loading from Bob's (denoted by *h*) that promised a delivering delay d_p within range R_ρ. Alice's PDA updates this probability at any time (*t-1*) (time at which it loads software from *h*) in two steps. First, it computes

$$\pi_h^{(t-1)}(d_p \in R_\rho | d_e \in R_\gamma) = \frac{D_\gamma^\rho}{D_\gamma},$$

where $D\gamma$ is the number of times Alice's PDA has experienced a delay ($d_e \in R_\gamma$) with *h* (up till time (*t-1*)), and $D\gamma^\rho$ is the number of times Bob's device promised a delay d_p in the range R_ρ ($d_p \in R_\gamma$) among the times considered in $D\gamma$.

This gives the probability that *h* had promised a delay d_p within R_ρ, but actually delivered within R_γ. The lower the difference between what *h* actually did (R_γ) and what it promised (R_ρ), the more reliable (trustworthy) *h*.

Second, Alice's PDA updates the following expression according to Bayes' theorem $\forall\ \gamma \in [1, 3]$:

$$\pi_h^t(d_e \in R_\gamma | d_p \in R_\rho) = \\ = \frac{\pi_h^{(t-1)}(d_e \in R_\gamma) \cdot \pi_h^{(t-1)}(d_p \in R_\rho | d_e \in R_\gamma)}{\sum_{\sigma \in [1,3]} \pi_h^{(t-1)}(d_e \in R_\sigma) \cdot \pi_h^{(t-1)}(d_p \in R_\rho | d_e \in R_\sigma)},$$

This is Alice's PDA prediction that, at future time *t*, if *h* promises a delay within R_ρ, it will actually deliver within R_γ. MATE then uses such predictions for computing elementary utilities as subsection 5.2 shows.

If Alice's PDA has never interacted with Bob's, then its MATE initializes all the probabilities corresponding to Bob's according to a uniform distribution:

$$\pi^0(d_e \in R_\gamma) = \pi^0(d_p \in R_\rho | d_e \in R_\gamma) = \frac{1}{3},$$

$\forall\ \gamma, \rho \in [1, 3]$.

As the Bayesian revision is for discrete (and not continuous) random variables, we do not have to impose any prior distribution, as opposite to the previous approach in which we integrated MATE with a trust model and had to assume a normal delay distribution.

6 EXPERIMENT

We now describe the experiment we have carried out to evaluate MATE's Bayesian revision process.

Goal: To determine whether MATE increases its component loader's utility.

Simulated configuration: We simulate a configuration consisting of two component loaders (each with a different supplier selection method) and three component suppliers (one for each supplier behavioural model).

Supplier behavioural model: A component supplier belongs to one of the following three behavioural models: highly unreliable, unreliable, and reliable. All suppliers promise to deliver components within range R_1 (seamless range), but whether they do so depends on their behavioural models. A *highly unreliable* supplier always delivers within range R_3 (risky range); an *unreliable* one within R_2 (limited disruptions range); and a *reliable* one delivers depending on its load (component requests from other loaders). To see how load matters for a reliable supplier, consider an example in which Alice's PDA (loader) asks Bob's (supplier) a component. Bob's PDA promises to deliver the component within a certain time range, but it then receives *n* requests until Alice's accepts the offer. This causes

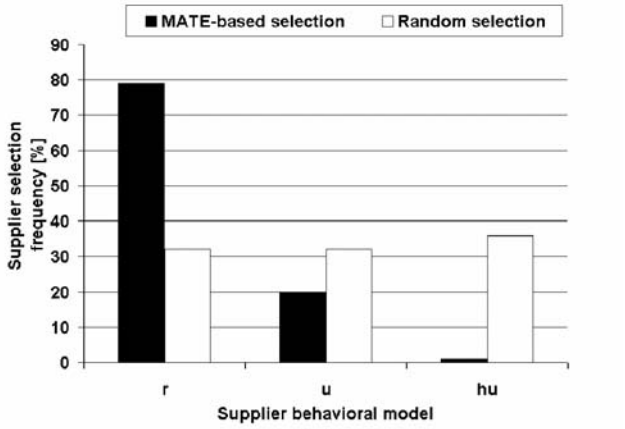

Figure 2 *Selection frequencies of suppliers by the component loader using random selection (empty bars) and by the component loader using MATE (filled bars). There is one supplier for each behavioural model: reliable (r), unreliable (u), and highly unreliable (hu).*

additional delay in Bob's PDA delivering the component to Alice's. We assume that if *n* <= 2, then Bob's PDA provides the component within R_1 (as promised), if n = 3 it provides within R_2, otherwise within R_3. We model the number of requests (*n*) as a Poisson probability distribution with an average μ = 2 (i.e., Bob's PDA delivers the component within R_1 on average). We choose such a distribution because it is often used to model number of arrivals (requests) that are randomly distributed in time, as in our case.

Supplier selection method: A component loader chooses a component supplier in two different ways. The first is random selection, i.e., the loader selects randomly the supplier. In the second way, it uses MATE (in the Bayesian version). For simplicity, MATE considers one preferable occurrence: that of absence of disruptions (*AD*). We intuitively set the corresponding positive contributions as follows: PC_{AD}(carry on seamlessly)=1, PC_{AD}(carry on with limited disruptions)=0.7, PC_{AD}(give up)=0, and PC_{AD}(user interacts with GUI)=0.4.

Experiment Metrics: We measure each component loader's overall utility, i.e., sum of its elementary utilities during the entire simulation duration.

Simulation execution: The experiment duration is of 100 time units. At each time unit, one component loader randomly chooses its supplier, and the other one uses MATE to select both action and supplier (i.e., it chooses the best action and, if the action consists in loading software, it also chooses the best component supplier). Based on the supplier's reliability (i.e., providing software timely), MATE then updates its trustworthiness in the supplier as

subsection 5.3 shows. We run the experiment 10 times and the results of all runs are averaged.

Results: The loader using MATE increases its overall utility by 32% than the loader using random selection. The reason is that MATE mostly (in the 79% of the cases) is able to select the best action and the most reliable component supplier at each time unit, as Figure 2 shows.

7 DISCUSSION

This section discusses how MATE compares with traditional assurance-based (security) mechanisms and then points out MATE design's limitations.

7.1 Assurance vs. trust

Social uncertainty produces risks, and either assurance or trust reduces it. Mehr and Cammack (1961) argued that risk is "uncertainty about loss". Similarly, Greene and Trieschmann (1962) defined risk as "uncertainty as to the occurrence of an economic loss". Knight (2002) defined risk as "measurable uncertainty". Clearly, most of these definitions have one thing in common: they all equate risk with *uncertainty*. Yamagishi and Yamagishi (1994) discussed two ways that reduce social uncertainty: *assurance* and *trust*. They argued that Japanese citizens have often lower level of trust compared with their American counterparts.

Not generalized trust, but rather mutual assurance characterizes Japanese society. Mutual assurance builds up on close, stable, long-term social relations; the sense of stability then reduces social uncertainty. In contrast, American society does not provide that high-degree of perceived stability and reduces social uncertainty relying on trust-related aspects, such as personalized knowledge and reputation information about others.

Roaming and pervasive devices cannot rely on stability: they need to cope with an enormous number of interactions, mostly with unknown devices, within network boundaries which possibly they have never experienced; these interactions may then lead to unpredictable outcomes.

In pervasive computing, the lack of stability leads to reduce uncertainty and associated risks mainly through trust-related technologies, though assurance-based mechanisms are still available in some cases. So, for example, devices controlling actuators in luxury cars may rely on *assurance*, i.e., they may only download software components provably authored by the car manufacturer (e.g., using public key encryption). In contrast, some PDA users currently install shareware and freeware with no confidence that the software performs as expected, since a trusted third party is not always available or even defined (they face risks without any protection mechanism). Between these two extremes, there remains room for modelling trust-informed decision-making in less-than-certain environments. Ours is one of such models as it is a well-founded decision model which selects the best component supplier device in the absence of assurance. Two main aspects contribute in reducing risks: (i) the device downloads frequently used software only from fully accredited systems and then keeps them stored; (ii) trust mechanisms, such as personal experiences and reputation information about component suppliers, feed the decision-making process.

7.2 Limitations

When we consider MATE integrated with a trust model, open issues that relate to trust models surface. For example, one of such issues is the ability to bind trust information to identities, in support of the formation and evolution of reputation and trust profiles. To than end, we assume that each mobile device has a public key, which represents its identity (or pseudonym). This, however, raises two concerns.

The first is to balance privacy with the ability to identify devices. Each single mobile device can create multiple public keys, thus being able to assume different identities (pseudonyms). The ease of creating and deleting pseudonyms can somewhat alleviate privacy concerns in ubiquitous environments (Seigneur and Jensen, 2004), but it is also an effective tool for malicious devices to repeatedly misbehave without being identified (Friedman and Resnick, 2001). To make creating pseudonyms harder, Quercia *et al.* (2006) proposed a scheme based on blinded threshold signature in which collections of devices certify pseudonyms, whilst protecting user anonymity. This scheme is not, however, a solution for all privacy-related problems but anonymity. For example, it still suffers from privacy breaching, e.g., a device reveals its user private data to a restricted and authorized group which then unjustly reveals it to unauthorized parties. Ahmed *et al.* (2005) proposed a scheme based on statistical traceability which aims at detecting privacy breaching.

The second concern stems from enabling identification of resource constrained devices (e.g., sensors, embedded systems), which cannot support public key cryptography. As a consequence, their pseudonyms cannot be public keys. In this case, we envision that a public key identifies the base station (a central device gathering data from its sensors) which shares symmetric keys with its sensor nodes.

Also, another limitation of MATE lies in the lack of detection mechanisms for malicious components. After experiencing damages, a device has to detect the responsible software component(s) so that MATE reliably updates the component supplier's reputation information.

8 FUTURE WORK

Dickson and Giglierano (1986) considered entrepreneurial risk as either the potential to act too quickly on an unsustainable opportunity, thus "sinking the boat", or the potential to wait too long before acting, thus "missing the boat". To adapt Dickson and Guglielmo's nautical analogy to our scenario, component loader devices may either incorrectly weigh up decisions, load software from malicious devices and expose themselves to security threats, or rely on devices that provide desired software too late to be useful. We considered only part of the equation, that of "missing the boat". We propose to extend our solution to situations that allow reasoning about both risks of out-of-time delivery and security threats that relate to mobile software components (Jansen and Karygiannis, 1999), thus avoiding "sinking the boat". How additional aspects may affect our model would be also of interest. For example, we have not considered that the decision of whether a software component will be locally stored or whether it will be mobile (i.e., loaded only when needed) depends on the required security level and on the sensitivity of the software component's task (Jansen and Karygiannis, 1999).

9 CONCLUSION

We have presented MATE, a conceptual model of decision-making running on a pervasive device. We have assumed that the device's software architecture is component-based (such as SATIN (Zachariadis et al., 2004)) and has discovery mechanisms (such as RUBI (Harbird et al., 2004)). Having a component-based model and a discovery mechanism, MATE is able to decide whether to load software components - it combines device user's risk attitudes, context of interaction, and software sources' trustworthiness so to apply the expected utility theory and consequently determine the best action with each source. As a consequence of the expected-utility decision rule, the device loads software only from trustworthy sources (i.e., those that have provided software mostly within the promised time frame) thus isolating untrustworthy ones.

The ideas behind MATE hold not only for trusted software component loading, but apply to any collaborative application which benefits from trust-informed decisions, thus making MATE a general decision module for any trust model. Generally, a collaborative application needs to take the decision whom to collaborate with. A trust model and MATE may support the decision: the trust model manages collaborating peering devices' trustworthiness and provides it to MATE, which then decides whether to collaborate and, if so, with whom.

ACKNOWLEDGEMENT

We thank Licia Capra for her comments and gratefully acknowledge the support of the European Commission through the SEINIT and RUNES projects.

REFERENCES

Abdul-Rahman, A. and Hailes, S. (2000) 'Supporting Trust in Virtual Communities'. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6, page 6007, Washington DC, USA, 2000. IEEE Computer Society.

Ahmed, M., Quercia, D. and Hailes, S. (2005) 'Statistical Matching Approach to Privacy Disclosure for Ubiquitous Computing Environment'. In *Proceedings of the 1st International Workshop on Trust, Security and Privacy for Ubiquitous Computing*, Taormina, Italy, June 2005. IEEE.

Bernoulli, D. (1954) 'Exposition of a new theory on the measurement of risk'. Econometrica, 22:2236, January 1954.

Blaze, M. Feigenbaum, J. and Keromytis, A.D. (1998) 'KeyNote: Trust Management for Public-Key Infrastructures'. In *Proceedings of the 6th International Workshop on Security Protocols*, volume 1550 of Lecture Notes in Computer Science, pages 59--63, Cambridge, UK, April 1998. Springer-Verlag.

Blaze, M., Feigenbaum, J. and Lacy, J. (1996) 'Decentralized Trust Management'. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 164--173, Oakland, CA, May 1996.

Cahill, V., Gray, E., Seigneur, J.-M., Jensen, C., Chen,Y., Shand, B., Dimmock, N., Twigg, A., Bacon, J., English, C., Wagealla, W., Terzis, S., Nixon,P., Serugendo, G., Bryce,C., Carbone, M., Krukow, K.and Nielsen, M (2003) 'Using Trust for Secure Collaboration in Uncertain Environments'. *IEEE Pervasive Computing Mobile and Ubiquitous Computing*, 2(3):5261, August 2003.

Capra, L.(2004) 'Engineering human trust in mobile system collaborations'. In *Proceedings of the 12th International Symposium on Foundations of Software Engineering*, pages 107--116, Newport Beach, CA, USA, November 2004. ACM Press.

Carbone, M., Nielsen, M., and Sassone, V. (2003) 'A Formal Model for Trust in Dynamic Networks'. In *Proceedings of the 1st International Conference on Software Engineering and Formal Methods*, pages 54--63, Brisbane, Australia, September 2003. IEEE.

Cervantes, H. and Hall, R.S. (2000) 'Beanome: A Component Model for the OSGi Framework'. In *Proceedings of the Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices*, Lausanne, Switzerland, September 2000.

Cervantes, H. and Hall, R.S. (2004) 'Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model'. In *Proceedings of the 26th International Conference on Software Engineering*, pages 614--623, Edinburgh, United Kingdom, May 2004. IEEE Computer Society.

Chen, Y., Jensen, C.D., Gray,E. and Seigneur, J. (2003) 'Risk Probability Estimating Based on Clustering'. In *Proceedings of the 4th IEEE Annual Information Assurance Workshop*, pages 229--233, United States Military Academy, West Point, New York, USA, June 2003. IEEE Systems, Man and Cybernetics Society.

Clarke, M., Blair,G.S., Coulson,G. and Parlavantzas, N. (2001) 'An Efficient Component Model for the Construction of Adaptive Middleware'. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, volume 2218 of Lecture Notes in Computer Science, pages 160--178, London, UK, 2001. Springer-Verlag.

Dickson, P. and Giglierano, J. (1986) 'Missing the boat and sinking the boat: a conceptual model of entrepreneurial risk'. J. Mark 50(7):5870, 1986.

Dimmock, N. (2003) 'How much is enough? Risk in Trust-based Access Control'. In *Proceedings of the 12th International Workshop on Enabling Technologies*, page 281,Washington, DC, USA, June 2003. IEEE Computer Society.

Dimmock, N., Bacon,J., Ingram, D. and Moody, K. (2005) 'Risk models for trust-based access control'. In *Proceedings of the 3rd Annual Conference on Trust Management*, volume 3477 of Lecture Notes in Computer Science, pages 364--371. Springer-Verlag, May 2005.

English, C., Terzis,S. and Wagealla, W. (2004) 'Engineering Trust Based Collaborations in a Global Computing Environment'. In *Proceedings of the 2nd International Conference on Trust Management*, volume 2995 of Lecture Notes in Computer Science, pages 120--134, Oxford, UK, March 2004. Springer.

Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H. and Yegin, A. (2003) 'Protocol for carrying authentication for network access (PANA)'. InternetDraft, Internet Engineering Task Force, July 2003.

Friedman, A. and Resnick, P. (2001) 'The social cost of cheap pseudonyms'. *Journal of Economics and Management Strategy*, 10(2):173199, 2001.

Greene, M.R. and Trieschmann, J.S. (1962) 'Risk and Insurance'. South-Western Publishing, Cincinnati, Ohio, 1962.

Harbird, R., Hailes, S. and Mascolo, C. (2004) 'Adaptive resource discovery for ubiquitous computing'. In *Proceedings of the 2nd Workshop on Middleware for pervasive and ad-hoc computing*, pages 155--160, Toronto, Ontario, Canada, October 2004. ACM Press.

Hirshleifer, J. and Riley, J.G. (1992) 'The Analytics of Uncertainty and Information'. Cambridge University Press, September 1992.

Jansen, W. and Karygiannis, T. (1999) 'Mobile Agent Security'. NIST special publication 800-19. National Institute of Standards and Technology, 1999.

Josang, A. and Presti, S.L. (2004) 'Analysing the Relationship between Risk and Trust'. In *Proceedings of the 2nd International Conference on Trust Management*, volume 2995 of Lecture Notes in Computer Science, pages 135--145, Oxford, UK, March 2004. Springer-Verlag.

Kent, S. and Atkinson, R. (1998) 'Security Architecture for the Internet Protocol'. RFC 2401, The Internet Engineering Task Force, November 1998.

Knight, F. 'Risk, Uncertainty and Profit'. Beard Books, Boston, 2002.

Levy, H. and Markowtiz, H.M. (1979) 'Approximating Expected Utility by a Function of Mean and Variance'. *American Economic Review*, 69(3):30817, 1979.

Marsh, S. (1994) 'Formalising Trust as a Computational Concept'. Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.

Mehr, R.I. and Cammack, E. (1961)'Principles of Insurance'. Richard D. Irwin, 1961.

Mui, L., Mohtsahemi, M. and Halberstadt, A. 'A Computational Model of Trust and Reputation'. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, page 188, Big Island, HI, USA, January 2002. IEEE Computer Society.

Neumann, J.V. and Morgenstern, O. 'Theory of games and economic behavior'. J. Wiley, New York, 1964.

Patrick, A. 'Building trustworthy software agents'. *IEEE Internet Computing*, 6(6):4653, 2002.

Quercia, D., Hailes, S. (2005) 'Risk Aware Decision Framework for Trusted Mobile Interactions'. In *Proceedings of the 1st IEEE/CreateNet International Workshop on The Value of Security through Collaboration*, Athens, Greece, September 2005.

Quercia, D.; Lad, M.; Hailes, S.; Capra, L.; Bhatti, S. (2006) 'STRUDEL: Supporting Trust in the Dynamic Establishment of peering coaLitions'. In *Proceedings of the 21st ACM Symposium on Applied Computing*. Dijon, France, April 2006.

Quercia, D.; Hailes, S.; Capra, L. (2006) 'B-trust: Bayesian Trust Framework for Pervasive Computing'. In *Proceedings of the 4th International Conference on Trust Management*, Lecture Notes in Computer Science, Pisa, Italy, May 2006. Springer-Verlag.

Quercia, D., Hailes, S., Capra, L. (2006) 'TATA: Towards Anonymous Trusted Authentication'. In *Proceedings of the 4th International Conference on Trust Management*, Lecture Notes in Computer Science, Pisa, Italy, May 2006. Springer-Verlag.

Savage, L. 'Foundations of Statistics'. John Wiley & Sons, New York, 1954.

Seigneur, J.-M. and Jensen, C.D. 'Trading Privacy for Trust'. In *Proceedings of the 2nd International Conference on Trust Management*, Lecture Notes in Computer Science, pages 93--107, Oxford, UK, March 2004. Springer-Verlag.

Wagealla, W., Carbone, M., English,C., Terzis,S., and Nixon, P. (2003) 'A Formal Model of Trust Lifecycle Management'. In *Proceedings of the 1st Workshop on Formal Aspects of Security and Trust*, September 2003.

Yamagishi, Y. and Yamagishi, M. (1994) 'Trust and Commitment in the United States and Japan'. *Motivation and Emotion*, 18(2):129166, 1994.

Zachariadis, S., Mascolo,C. and Emmerich, W. (2004) 'SATIN: A Component Model for Mobile Self-Organisation'. In *Proceedings of the International Symposium on Distributed Objects and Applications*, volume 3291 of Lecture Notes in Computer Science, pages 1303--1321, Agia Napa, Cyprus, October 2004. Springer-Verlag.