# Risk Aware Decision Framework for Trusted Mobile Interactions

Daniele Quercia and Stephen Hailes
Department of Computer Science, University College London, London, WC1E 6BT, UK.
{D.Quercia, S.Hailes}@cs.ucl.ac.uk

## Abstract

*Adaptation to context is likely to be a key element in ensuring that pervasive devices make the most efficient use of the limited resources available to them. This adaptation can occur on different timescales: from very rapid adaptation to network congestion, through software component discovery and loading to allow for the addition of new functionality, to longer-term component update and patching. In the latter two cases, dynamic code loading introduces security problems, particularly because it may be from untrusted sources with whom pervasive devices happen to be networked at the time of need. As a consequence, we propose a local decision-making process that aims at producing better-informed decisions for pervasive devices when they contemplate whether or not to load software from other devices. This process has three key elements: (i) explicit identification of potential risks, given the device's context and the type of application; (ii) computation of likelihoods with which the risks will occur, based on trust mechanisms; (iii) integration of the risk attitude of the user of the device, through customisable elementary utility functions.*

## 1 Introduction

Fostering the pervasive computing vision will necessitate the installation of a wide range of software components on mobile and even wearable devices. Pervasive (or ubiquitous) computing is a term used to describe a computational environment in which devices that are mobile and sometimes resource constrained (e.g., sensors, 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 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, they will have to rely on loading the components dynamically and as necessary from other devices.

Installation of software components requires additional security mechanisms and a strong degree of assurance that the components are as they claim to be - for example, that they are not viral and do not contain trojans. In traditional systems, this can be handled by digitally signing software and 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 to allow for appropriate adaptation 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.

Computational agents based on the concept of trust have been extensively researched as security frameworks 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, have been found to be reliable in past software exchanges – they have always cooperated supplying what they were expected to. Trust-based decisions can be enhanced if they explicitly consider potential risks. In fact, as supporting evidence, we argue that 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 [31]: 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 is likely to 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 that has been published in the field of the integrated management of trust and risk is rather general and, as a consequence, cannot be directly applied to deal with 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.

Our research goal is to realise the full potentials of ubiquitousness of mobile software components. Thus, we present a novel decision framework, that integrates both risk and trust. This allows mobile devices to reason about exchanging software components under uncertainty. It possesses a range of appropriate properties: (i) explicit modeling of decision making in the absence of complete information; (ii) integration of the actual risk attitude of the user of the mobile device, by means of a customizable utility function; (iii) clear computation of likelihoods of the potential risks, based on trust mechanisms.

The remainder of the paper will be focused on elucidating the details of a model for decision-making under uncertainty. Section 2 discusses related work. In section 3 we introduce a scenario intended to be illustrative of a real-life usage of the model. In section 4 we describe all elements modeling the decision problem space. Section 5 discusses a simple quantitative application of the model. In section 6, we make a clear distinction between trust and assurance and we then discuss in depth our solution. Finally, section 7 describes future work and section 8 presents our conclusions.

## 2 Related Work

The need to deal with trust and reputation has long been realised in the pervasive computing research community, and many approaches have been proposed. The notion of trust management has formed a subject of study since work by Blaze, Feigenbaum, and Lacy with their seminal paper on decentralised trust management [5], which shows a language for specifying trusted actions and trust relationships; they also describe a prototype implementation of a trust management system, called PolicyMaker. KeyNote [4], a similar credential-based distributed policy management framework, has then been introduced: it puts specific emphasis on access control decisions rather than general trust management (e.g., it does not address trust evolution issues). In computer science, Marsh was the first one to introduce a computational model for trust [27]. Abdul-Rahman and Hailes [1] then proposed a distributed trust management model, with which entities could autonomously reason about trust, without relying on a central authority. Mui [29] then investigated a computational model of trust, in which the concept of reputation was introduced. More recently, Capra [7] proposed a human trust management model, specifically for mobile systems and applications. This framework tries to capture the actual human disposition to trust of the user of the mobile device, through customizable functions.

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 [11]. The SECURE project [6] incorporated a formal trust model [8] and an explicit model of risk [14] 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.* [34] 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 [14] 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 certainties of events (i.e., only when an event takes place can its utility be evaluated).

More recently, trust-based risk investigation has led to the integration of the expected utility theory in computational risk models. Jøsang and Lo Presti [23] 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 [16] and, thus, integrate the trust assessment within a global risk-aware decision framework. Dimmock *et al.* [15] investigated a model of risk again based on the expected utility theory. This has been evaluated through 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.

## 3 Scenario

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 [9], Gravity [10], SATIN [36] and OpenCOM [12]. We also assume that components are available together with their dependencies.

Let us consider the following scenario. 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 realises 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 a result of 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 [17]) and a software component for confidentiality protection (e.g., IPSec [24]).

*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.

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 organisations or individuals that authored the components. Thus, when the component loader needs a software component $C$ (e.g., IPSec), it attempts to discover nearby devices (component suppliers) that are willing to provide a component with the requisite semantics and within the timeframe 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 timeframe requested. The component loader uses our decision framework to select which of the component suppliers to use, based on the likelihoods 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.

We do not deal with the aspect of component discovery. For this purpose, any appropriate discovery framework available in the literature can be applied; for example, RUBI [20] is a resource discovery framework for mobile devices in which discovery is based on devices' local views of the network structure.

In the next section, we will discuss our scheme which represents a decision framework for component loaders. 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 should be able to take 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 Risk-aware Decision Framework

In economics, decision theory under uncertainty is often realised in terms of expected utility (EU) theory. Bernoulli [3] first suggested that people maximise not the expected monetary *value* of an outcome, but its *expected utility*. This can be represented as

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

where $EU$ is the expected utility, $\pi_i$ is the probability that the $i^{th}$ outcome will occur and $u(o_i)$ is the utility of the $i^{th}$ outcome. Furthermore, Bernoulli argued that the psychological value of money diminishes as its quantity increases. He thus represented the utility function, which shapes the functional relation between elementary utility and value of outcomes, with a logarithmic form. After that, Von Neumann and Mongestern [30] showed that the maximization of expected utility is a rational behaviour, given the validity of 6 axioms describing the rational behaviour of decision makers. This theory assumes that the decision-making process is supported by the *objective* probability that an outcome occurs. Amongst others, Savage [32] developed a version of EU theory in which probabilities are subjective.

[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.

Our decision framework is based on EU theory and has the following properties: (i) outcome probabilities are not objective, but are considered to be subjective, as they depend, as we will see, on subjective evaluations such as trust values; (ii) utility functions are not elicited by means of conventional methods used in economics (e.g., lottery technique [21]), but are customised according to the risk attitude of the user of the component loader device.

The remainder of this section will focus on elucidating our framework's building blocks. First, the elements of decision under uncertainty are introduced. Second, the overall utility assessment for a single action is shown. Third, the elementary utility function for an outcome is presented. Finally, the functional relation between likelihood of outcome states and trust levels is illustrated.

## 4.1 Elements of Decision Under Uncertainty

The component loader bases its decision whether to take the component $C$ from any component supplier on the following elements [21]:

1. a set of *actions* $(a_1, \ldots, a_x, \ldots, a_A)$ that can be carried out by the component loader. For example, possible component loader's actions include *'accept component'*,*'do not accept component'* and *'ask the user for further guidance'*;

2. a set of *states* $(s_1, \ldots, s_y, \ldots, s_S)$ of the world that model the uncertainty experienced by the component loader when interacting with any component suppliers. This set of states completely defines the set of occurrences that are 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'*;

3. an *outcome* function $o(a_x, s_y)$ that returns the outcome if the component loader carries out action $a_x$ and the world's state is $s_y$. For example, the combination of the action *'no component accepted'* and any possible states of the world 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., the confidentiality of a video-conference can be only assured with a piece of code that performs a robust encryption algorithm; if such piece of software is neither locally stored nor remotely available, the video-conference must be stopped);

4. a *probability function* $\pi(s_y)$ that expresses component loader's belief that state $s_y$ will occur. For example, if the component loader believes that the component supplier is very trustworthy, it may intuitively assign a very low probability to the occurrence of the state *'component supplier delivers a malicious component'*, whereas the state *'component supplier successfully delivers the component'* will be believed to be more likely. Similar considerations apply to the trust level of the component author: if the integrity of the software component is guaranteed and the component-loader highly trusts the component author, then the likelihood of the state *'component supplier delivers a malicious component'* is close to the minimum value;

5. an *elementary-utility* function $u(value(o_z))$ that measures the desirability of the any given outcome value $value(o_z)$ to the component loader. Within this section, we will show the construction of such function.

## 4.2 Decision Rule

The component loader applies the expected-utility rule to decide upon the most advantageous action. Given a set of possible actions such as *'carry on the video-conference'* or *'give up the video-conference'*, the component loader derives an ordering of such actions, by means of 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 $a$ is

$$U_a^h = \sum_{i \in [1,S]} \pi^h(s_i) \cdot u(v(o(a, s_i))),$$

where $\pi^h(s_i)$ is the probability that state $s_i$ will occur, given that $h$ is the component supplier; if $s_i$ occurs, the value of such outcome attached by the component loader is denoted by $v(o(a, s_i))$. For a given action, the component loader maximise $U_a^h$ 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.

We now introduce the two composing elements of the utility assessment for an action: first, the elementary utility function; second, the probability function.

## 4.3 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, preference elicitation is done by means of a policy refinement process: from high-level user policy specifications the component loader device extracts the appropriate parameters (denoted by $w_i$) to attach a number ($value(o)$) to each single outcome $o$ (Table 1 shows a list of possible outcomes). In so doing, the risk attitude of the user of the component loader device is considered. The elementary utility function then statically maps this value into its *utility*.

We first describe the construction of the elementary utility function $u : value(o) \rightarrow u(value(o))$. We then explore how the component loader quantitatively estimates $value(o)$ for any outcome $o$. The elementary utility function converts the user-defined value of an outcome under certainty to a level of satisfaction (utility). We consider the shape of the elementary utility function logarithmic: that is, as shown by Bernoulli, user's attitudes are predominately risk-averse. The elementary utility function has then the following form:

$$q(x) = log(1 + x), \tag{1}$$

where $x$ is the value of the outcome under consideration. To enhance computational tractability on mobile devices, the above elementary utility function can be simplified with a mean-variance analysis, a method currently used for risk assessment on financial portfolios. Levy and Markowitz [26] showed that a simpler mean-variance approximation yields a level of utility almost equal to that obtained by the logarithmic utility function. Put simply, widely used utility functions, including expression (1), can be approximated as second degree Taylor polynomials, whilst ensuring compatibility with the expected utility theory. The second degree Taylor Polynomial of a utility function $v(x)$ at $x_0$ can be written as

$$v(x) = v(x_0) + v'(x_0)(x - x_0) + \frac{v''(x_0)}{2}(x - x_0)^2 \tag{2}$$

If function $v$ has been chosen such that $v(x_0) = 0$ and $v'(x_0) = 1$, we then have

$$v(x) = (x - x_0) + \frac{v''(x_0)}{2}(x - x_0)^2 \tag{3}$$

For $x_0 = 0$ (small losses), expression (1) satisfies the conditions $q(x_0) = 0$ and $q'(x_0) = 1$. It can be thus written in a form similar to expression (3):

$$q(x) \approx (x - 0) + \frac{q''(0)}{2}(x - x_0)^2 = x - \frac{1}{2}x^2 = u(x) \tag{4}$$

Our decision framework will use function $u$ as elementary utility function, within range [0,1]. This range assures the validity of Taylor approximation, as it is close to point $x_0 = 0$.

There now remains one thread to unwind: that of how the component loader quantitatively estimates $value(o)$ for any outcome $o$. In order to attempt this, we need to define some new concepts.

Any application is characterised by particular quality of service parameters, the importance of which changes as the context of operation changes. For example, aspects associated with a video-conference application might include assurances about confidentiality or authenticity, and resistance to disruption or termination of the conference. We call these aspects *application dimensions*.

For each application dimension, we need to understand the importance that the component loader associates with it. Thus, for example, confidentiality generally has medium importance, and disruption or termination of the conference usually represent a prime concern. In the following, we use $w_i$ to represent the value that the component loader attaches to the $i^{th}$ application dimension. This is intended to reflect the preferences of the component loader's user. User preferences are stored as high-level policy on the component loader device, which then performs a policy refinement process: it extracts from high-level policy specification each $w_i$ factor. For example, the user specifies by means of a GUI how much he cares about disruptions, and the component loader device extracts a value for $w_{AD}$, which represents the preference of the user in always having absence of disruptions (AD).

In addition to this, we should be able to estimate the degree to which each application dimension is important when both a specific type of application and a particular outcome are considered. Thus, in the video conferencing scenario, this function might include the following elements:

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

2. *spared user time* ($D_{SUT}(o)$) – Outcomes causing massive user intervention bring this value down to the minimum 0, whereas maximum value 1 is only experienced when user interventions have not been required;

3. *inverse security gap ($D_{ISG}(o)$)* – To clarify the meaning of this dimension, let us imagine the following situation. A device 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 dimension 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. This dimension can be expressed as

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

Note that the dimension refers to general security; however, it would be more appropriate to have one dimension for each specific security aspect, such as confidentiality, integrity and authentication.

Finally, we can build the value of outcome $o$ under certainty, which always falls in the range $[0, 1]$, as

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

where $w_i$ is the component loader's value attached to the $i^{th}$ application dimension and $D_i(o)$ is the value of the $i^{th}$ application dimension in case outcome $o$ occurs. The value of outcome $o$ is then used as input for the elementary utility function $u$.

### 4.4 Probability Function

The *probability function* expresses component loader's belief that a certain state will occur when interacting with a specific component supplier. For example, Alice's PDA acting as component loader computes the likelihood of state *'component supplier delivers the component on time'* when interacting with a specific component supplier, say Bob's PDA.

We believe that our decision framework is best considered as integrated with a distributed trust model which returns component suppliers' and component authors' trustworthiness, that then contributes to the computation of state probabilities. Thus, for example, the likelihood of a state, say *'component supplier delivers a malicious component'*, depends on both component supplier's and component author's trustworthiness as well as the guarantees that accompany the agent about the integrity and authenticity of the component. The component loader utilises the past to illuminate the present: from past interactions, received recommendations, and reputation information it estimates trustworthiness [14]. Based on trustworthiness, the component loader estimates the probability of each state: as the trust level in component author increases, the probability, for example, of obtaining a malicious component decreases.

Part of section 5 discusses the relationship between trust and state probability in a specific situation.

## 5 Our Decision Framework in Action: a Detailed Example

For the purpose of simplicity, we restrict our discussion to risks that relate to the ability of the component supplier device to deliver a software component within a set period of time. 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.

Returning to our example, Alice's PDA needs a software component and must decide whether or not to accept it from a nearby device, say Bob's PDA. Let us assume that the latter device, the component supplier, has declared providing delay $d_p$ and confidence level $CL$ that the component will be transmitted on time. For example, Bob's PDA declares to be able to provide the software component with delay $d_p = 10$ seconds and with confidence level $80\%$. The delay that Alice's PDA actually experiences is denoted by $d_e$.

### 5.1 Elements of Decisions in our Scenario

As in section 4.1, the decision problem space of Alice's PDA is modeled by the following elements:

1. a set of actions $(a_1, \ldots, a_x, \ldots, a_A)$, as listed in the first column of Table 1;

2. a set of states $(s_1, \ldots, s_y, \ldots, s_S)$. One state could be that the actual delay experienced by Alice's PDA in receiving the component from Bob's PDA falls in a range that allows the video-conference to continue seamlessly. For the purpose of the scenario, we define three possible ranges of delay $d_p$: the seamless range ($R_1$), the limited disruptions range ($R_2$) and the risky range ($R_3$). The first row of Table 1 lists this set of states;

3. an outcome function $o(a_x, s_y)$, which is the set of outcomes depending on both Alice PDA's action $a_x$ and state of the world $s_y$. Table 1 is the outcome matrix and shows the outcomes corresponding to all combinations of action and state;

4. a probability function $\pi(s_y)$ which expresses Alice PDA's belief that state $s_y$ occurs. For instance, $\pi(d_e \in R_1)$ is the probability that Bob's PDA delivers the component with a delay $d_e$ that falls in the seamless range, denoted by $R_1$. The probability of a state

depends on the declared delay, on the confidence level and on the trust that Alice's PDA has in Bob's PDA in providing the software component within the specified time. Further aspects associated with the probability function are discussed in subsection 5.2.

5. an elementary utility function $u(o_z)$, which measures the desirability of the different outcomes to Alice's PDA. Subsection 4.3 discusses in depth the function used and the factors affecting it.

| **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 |

**Table 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 Probability Function

Given a specific state (e.g., one amongst the states listed in the first row of Table 1, such as '$d_e \in R_1$'), this function returns the probability of its occurrence. Let $f(x)$ be the density function for the delay $d_e$ (assumed to be random), which is the delay that Alice's PDA will actually experience if Bob's PDA delivers the component. The probability that $d_e$ falls in, say, the seamless range $R_1$ is the integral of $f$ on that range; in other words:

$$\pi(d_e \in R_1) = \int_{R_1} f(x)dx. \tag{5}$$

The probabilities for the remaining ranges can be similarly computed. For the purpose of this paper, we suppose that $X$ is a normal distribution. The choice of the normal distribution is intended 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, the normal distribution is described by two terms, the mean $\mu$ and its standard deviation $\sigma$. X's mean depends on the trust level $T$ (subjective belief) that Alice's PDA has that Bob's PDA delivers the software on time. For the maximum level of trust, we set the mean equal to $d_p$. As the trust level decreases, the mean increases up to a worst case delay value, experienced when $T$ reaches the minimum level (complete distrust). Since $\mu$ has now been computed, to calculate expression (5) 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 arise: (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:

$$P(d_e \in U^+(\mu)) = \alpha T + \beta CL, \tag{6}$$

where $\alpha$ and $\beta$ are positive constants, such that $\alpha + \beta = 1$. In other words, the probability that the delay experienced falls in a range close to the mean, denoted by the right neighborhood $U^+(\mu)$ of mean $\mu$, solely depends on both trust level $T$ and confidence level $CL$. To define the right neighborhood of $\mu$, we choose a small positive real number $\delta$ so that $U^+(\mu) = \{x \in \mathbb{R} | 0 < x - \mu < \delta\}$. We are not interested in the left neighborhood because we only consider 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 neighborhood $U^+(\mu)$ 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 $\mu$ and variance $\sigma^2$, we can define $Z = \frac{X-\mu}{\sigma}$, which is the *standard* normal distribution, that is, a normal distribution with mean equal to 0 and unitary standard deviation. We can then write:

$$\begin{aligned} P(d_e \in U^+(\mu)) &= P(\mu < X < \mu + \delta) = \\ &= P(0 < Z < \frac{\delta}{\sigma}) = \\ &= \phi_Z(\frac{\delta}{\sigma}) - \phi_Z(0) = \\ &= \phi_Z(\frac{\delta}{\sigma}) - \frac{1}{2}, \end{aligned} \tag{7}$$

where $\phi_Z$ is the cumulative distribution function of standard normal distribution $Z$.

We then combine equation (6) and equation (7) to obtain the functional dependency of X's standard deviation $\sigma$ on the trust level and on the confidence level:

$$\sigma = \frac{\delta}{\phi_Z^{-1}(\alpha T + \beta CL + \frac{1}{2})}. \tag{8}$$

Given trust level $T$ and confidence level $CL$, the probability of each state can be computed. Equation (8) gives standard deviation of $X$, which has a known mean: it depends on $d_p$, the providing delay declared by Bob's PDA. Having X's mean and X's standard deviation, equation (7), which computes the probability of a given state, is easily evaluated. In fact, having $\mu$ and $\sigma$, $X$ can be converted into the standard normal distribution $Z$, which has been extensively tabulated and, thus, the computation of the integral (5) can be done with simple table lookups.

## 6 Discussion

Social uncertainty produces risks and is reduced mainly by means of assurance and trust. Mehr and Cammack [28] argued that risk is 'uncertainty about loss'. Similarly, Greene [19] defined risk as 'uncertainty as to the occurrence of an economic loss'. Knight [25] defined risk as 'measurable uncertainty'. Clearly, most of the definitions quoted above have one thing in common: they all equate risk with *uncertainty*. Yamagishi and Yamagishi [35] discussed two ways in which social uncertainty is reduced: *assurance* and *trust*. They argued that Japanese citizens have often lower level of trust compared with their American counterparts. Japanese society is characterised not by generalised trust, but rather by mutual assurance. This mutual assurance is based on close, stable, long-term social relations. Social uncertainty is reduced because of this sense of stability. In contrast, American society does not provide that high-degree of perceived stability and social uncertainty is thus reduced relying on trust-related aspects, such as personalised knowledge and reputation information about others.

It appears very unlikely that roaming, pervasive devices can rely on a sense of stability: they need to be able to cope with an enormous number of interactions, mostly with unknown devices, within network boundaries which possibly have never been previously experienced; these interactions may lead to unpredictable outcomes. Hence the need in pervasive computing to reduce uncertainty and, consequently, associated risks mainly by means of trust-related technologies, though assurance-based mechanisms can be still available in some cases. So, for example, devices controlling actuators that are embedded in luxury cars would need to rely on a *full level of assurance* and, thus, they only download software components provably authored by the car manufacturer. In contrast, some PDA users currently install shareware and freeware with little assurance that the software performs as expected, since a trusted third party is not always available or even defined. Between these two extremes, there remains room for systems that model autonomous decision-making processes in less-than-certain environments. Our model represents an advancement towards such systems, in terms of providing a rigorous decision-framework to select the best component supplier device in the absence of full assurance. There are, however, some aspects in the decision framework which increase the level of assurance: (i) frequently used software is downloaded from fully accredited systems and is then kept stored on the device; (ii) trust mechanisms, such as personal experiences and reputation information about component suppliers, are integrated in the decision-making process.

One of the main issue that distributed trust models should address is the ability to bind trust information to identities, in support of the formation and evolution of reputation and trust profiles. For the purpose of this paper, 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 [33], but it is also a nice tool for malicious devices to repeatedly misbehave without being identified [18]. This issue should partially affect the way in which trust in newcomers is bootstrapped. However, there would still be a tension between the preservation of privacy through partial anonymity and identification that is necessary both in order to detect malicious behaviors and to provide foundation for trustworthiness assessments. Statistical traceability [2] can act as a basis for reaching a proper balance. 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 the base station, a central device gathering data from its sensors, is identified by a public key and shares symmetric keys with its sensor nodes.

We do not deal with another issue which relates to trust management, that of trust dissemination: how recommendations and other types of trust-related information (e.g., credentials) are disseminated. Despite the inadequacies of our proposal for addressing trust dissemination, we do strike a resonant chord on the importance of minimizing the amount of disseminated trust information. To find partial solution to the problem, we may turn to [7], which proposes a protocol for the dissemination of minimum information upon which mobile devices can assess trustworthiness of other devices.

## 7 Future Work

Dickson and Giglierano [13] 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. In section 5, we considered only part of the equation, that of ‘missing the boat’. We propose to extend our short exposition here to apply the risk-aware decision framework to situations that allow both reasoning about risks of out-of-time delivery and security threats that relate to mobile software components [22], thus avoiding ‘sinking the boat’.

It would be also interesting to understand how our framework will be affected if new factors come into play. For example, it is very well to argue 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 level of security for the application and the sensitivity of the software component’s task [22]. Another research direction would be to look at how the sensitivity of the software component’s task can be integrated in the computation of the elementary utility function.

Since it is uncertain whether user preferences could be modeled with risk-averse logarithmic functions, we plan to use a risk-neutral function and then separately affect the utility computation with a factor modeling user risk attitudes.

Another assumption can be relaxed: that of the normal distribution for the density probability function. Baeysian statistics would be an effective tool for modeling state probability functions with few prior assumptions about their density distributions.

## 8 Conclusion

We have presented a conceptual model of decision-making for pervasive devices, which load software components from other devices when needed. The model is able to make decisions on whether to load software in the absence of complete information, by means of mechanisms which integrates trust within a general risk-based framework. Moreover, the actual risk attitude of the user of the mobile device is considered in the decision process, through customisable elementary utility functions.

## 9 Acknowledgements

The authors wish to thank Licia Capra for her useful comments and gratefully acknowledge the support of the European Commission through the SEINIT and RUNES projects.

## References

[1] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In *Proceedings of the* $33^{rd}$ *Hawaii International Conference on System Sciences*, volume 6, page 6007, Washington DC, USA, 2000. IEEE Computer Society.

[2] M. Ahmed, D. Quercia, and S. Hailes. Statistical Matching Approach to Privacy Disclosure for Ubiquitous Computing Environment. In *Proceedings of the* $1^{st}$ *International Workshop on Trust, Security and Privacy for Ubiquitous Computing*, Taormina, Italy, June 2005. IEEE.

[3] D. Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22:22–36, January 1954.

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

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.

[6] V. Cahill, E. Gray, J.-M. Seigneur, C. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen. Using Trust for Secure Collaboration in Uncertain Environments. *IEEE Pervasive Computing Mobile and Ubiquitous Computing*, 2(3):52–61, August 2003.

[7] L. Capra. Engineering human trust in mobile system collaborations. In *Proceedings of the* $12^{th}$ *International Symposium on Foundations of Software Engineering*, pages 107–116, Newport Beach, CA, USA, November 2004. ACM Press.

[8] M. Carbone, M. Nielsen, and V. Sassone. A Formal Model for Trust in Dynamic Networks. In *Proceedings of the* $1^{st}$ *International Conference on Software Engineering and Formal Methods*, pages 54–63, Brisbane, Australia, September 2003. IEEE.

[9] H. Cervantes and R. S. Hall. 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.

[10] H. Cervantes and R. S. Hall. Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model. In *Proceedings of the* $26^{th}$ *International Conference on Software Engineering*, pages 614–623, Edinburgh, United Kingdom, May 2004. IEEE Computer Society.

[11] Y. Chen, C. D. Jensen, E. Gray, and J. Seigneur. Risk Probability Estimating Based on Clustering. In *Proceedings of*

*the* $4^{th}$ *IEEE Anual Information Assurance Workshop*, pages 229–233, United States Military Academy, West Point, New York, USA, June 2003. IEEE Systems, Man and Cybernetics Society.

[12] M. Clarke, G. S. Blair, G. Coulson, and N. Parlavantzas. 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.

[13] P. Dickson and J. Giglierano. Missing the boat and sinking the boat: a conceptual model of entrepreneurial risk. *J. Mark*, 50(7):58–70, 1986.

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

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

[16] C. English, S. Terzis, and W. Wagealla. Engineering Trust Based Collaborations in a Global Computing Environment. In *Proceedings of the* $2^{nd}$ *International Conference on Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 120–134, Oxford, UK, March 2004. Springer.

[17] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for carrying authentication for network access (PANA). Internet Draft (work in progress), Internet Engineering Task Force, July 2003.

[18] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[19] M. R. Greene and J. S. Trieschmann. *Risk and Insurance*. South-Western Publishing, Cincinnati, Ohio, 1962.

[20] R. Harbird, S. Hailes, and C. Mascolo. Adaptive resource discovery for ubiquitous computing. In *Proceedings of the* $2^{nd}$ *Workshop on Middleware for pervasive and ad-hoc computing*, pages 155–160, Toronto, Ontario, Canada, October 2004. ACM Press.

[21] J. Hirshleifer and J. G. Riley. *The Analytics of Uncertainty and Information*. Cambridge University Press, September 1992.

[22] W. Jansen and T. Karygiannis. Mobile Agent Security. NIST special publication 800-19. Technical report, National Institute of Standards and Technology, 1999.

[23] A. Jøsang and S. L. Presti. Analysing the Relationship between Risk and Trust. In *Proceedings of the* $2^{nd}$ *International Conference on Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 135–145, Oxford, UK, March 2004. Springer-Verlag.

[24] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, The Internet Engineering Task Force, November 1998.

[25] F. Knight. *Risk, Uncertainty and Profit*. Beard Books, Boston, 2002.

[26] H. Levy and H. M. Markowtiz. Approximating Expected Utility by a Function of Mean and Variance. *American Economic Review*, 69(3):308–17, 1979.

[27] S. Marsh. Formalising Trust as a Computational Concept. Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.

[28] R. I. Mehr and E. Cammack. *Principles of Insurance*. Richard D. Irwin, 1961.

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

[30] J. V. Neumann and O. Morgenstern. *Theory of games and economic behavior*. J. Wiley, New York, 1964.

[31] A. Patrick. Building trustworthy software agents. *IEEE Internet Computing*, 6(6):46–53, 2002.

[32] L. Savage. *Foundations of Statistics*. John Wiley & Sons, New York, 1954.

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

[34] W. Wagealla, M. Carbone, C. English, S.Terzis, and P. Nixon. A Formal Model of Trust Lifecycle Management. In *Proceedings of the* $1^{st}$ *Workshop on Formal Aspects of Security and Trust*, September 2003.

[35] T. Yamagishi and M. Yamagishi. Trust and Commitment in the United States and Japan. *Motivation and Emotion*, 18(2):129–166, 1994.

[36] S. Zachariadis, C. Mascolo, and W. Emmerich. 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.