Dublin Institute of Technology
ARROW@DIT

Conference papers

School of Computing

2008-06-01

Resolving Uncertainty in Context Integration and Abstraction

Juan Ye
University of St. Andrews

Susan McKeever
Dublin Institute of Technology, susan.mckeever@dit.ie

Lorcan Coyle
University College Dublin

Steve Neely
University College Dublin

Simon Dobson
University College Dublin

Follow this and additional works at: http://arrow.dit.ie/scschcomcon

Recommended Citation

Juan Ye, Susan McKeever, Lorcan Coyle, Steve Neely, and Simon Dobson (2008). Resolving Uncertainty in Context Integration and Abstraction. ICPS 2008: Proceedings of the International Conference on Pervasive Services, ACM, June 2008

This Conference Paper is brought to you for free and open access by the School of Computing at ARROW@DIT. It has been accepted for inclusion in Conference papers by an authorized administrator of ARROW@DIT. For more information, please contact yvonne.desmond@dit.ie, arrow.admin@dit.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

Resolving Uncertainty in Context Integration and Abstraction*

[Context Integration and Abstraction]

Juan Ye, Susan McKeever, Lorcan Coyle, Steve Neely and Simon Dobson
System Research Group, UCD
Dublin, Ireland
juan.ye@ucd.ie, susan.mckeever@ucd.ie, lorcan.coyle@ucd.ie, steve.neely@ucd.ie, simon.dobson@ucd.ie

ABSTRACT

Pervasive computing is typically highly sensor-driven, but sensors provide only *evidence of fact* rather than facts themselves. The uncertainty of sensor data will affect each component in a pervasive computing system, which may decrease the quality of its provided services. We provide a general model to represent semantics of uncertainty in different levels (e.g., sensor, lower-level context and higher-level context). Within our model, fine-grained approaches are applied to evaluate and propagate uncertainties. They will help to resolve the uncertainty in each process of context management so that the effect of uncertainty on system services will be minimised.

Categories and Subject Descriptors

D2.2 [**SOFTWARE ENGINEERING**]: Design Tools and Techniques; I2.4 [**ARTIFICIAL INTELLIGENCE**]: Knowledge Representation Formalisms and Methods

General Terms

Management

Keywords

Context-aware computing, Context integration, Context abstraction, Uncertainty, Bayes' Theorem

*This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 "Towards a semantics of pervasive computing", 04/RPI/1544 "Secure and predictable pervasive computing", and Enterprise Ireland under grant number CFTD 2005 INF 217a, "Platform for User-Centred Design and Evaluation of Context-Aware Services".

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICPS'08, July 6–10, 2008, Sorrento, Italy.
Copyright 2008 ACM 978-1-60558-135-4/08/07 ...$5.00.

1. INTRODUCTION

Pervasive computing aims to provide services that respond directly to their user and environment with minimal intrusiveness and inherent pro-activity. This is achieved by assuming a number of invisible sensing and computational devices in an environment, which collect information about users and the environment. With the help of these devices, a pervasive computing system can deliver customised services to users in a contextual manner. Data in pervasive computing environments may be generated by untrustworthy or inaccurate sources and so should be taken "with a grain of salt". Because components of a pervasive computing environment deal with the real world, they come with certain caveats: sensors in the field are inherently inaccurate, since they could break down; or they could report inaccurately because they come up against a phenomenon for which they have not been designed [26]. Uncertainty may have an influential effect on the quality of services that a pervasive computing system provides. Inaccurate sensor data may result in misunderstanding of a user's or an environmental state, which leads to incorrect behaviour. Therefore, the issue of resolving uncertainty must be taken into account when dealing with pervasive computing systems.

Context-awareness is an enabling technology for pervasive computing. A context-aware computing system exhibits appropriate and customised behaviours that adapt to the change of users' context. *Context* can be any information that is used to characterise the situation of service consumers, which includes information about consumers, their environment, or their tasks [8]. Context is acquired from various kinds of sensors that are distributed in a pervasive computing environment. It can be sensed from physical devices, profiled from users, or derived from application- or meta-information existing in systems [29]. The uncertainty in sensor data will be transferred to context uncertainty and propagated through all the processes of context management. The question is how to model the semantics of context uncertainty and how to resolve (or minimise) uncertainty by distilling the most accurate context from a large number of trivial and noisy contexts.

Our work aims to propose a fundamental model of context uncertainty that represents semantics of context uncertainty and exhibits fine-grained approaches to evaluate and resolve uncertainty when processing and using context. To accomplish this, we analyse the essential characteristics of context types and constituents of context values, based on which

different types of context uncertainty will be discussed: *out-of-date*, *incomplete*, *imprecise*, and *inaccurate* [13]. We will discuss how context uncertainty is acquired from sensor uncertainty and explore the uncertainty propagation issue in two processes of context management: *context integration* and *context abstraction*. Context integration is about extracting the most accurate context from a number of noisy and conflicting contexts. Context abstraction is about deriving a higher-level application-interesting contexts (for example, a user state being in a "meeting" or "working") from a number of lower-level contexts (such as a user's location or a temperature in a room).

The remainder of this paper is organised as follows. Section 2 investigates various sensor uncertainties in different types of sensors. Section 3 provides a general definition of context from the perspective of representing a context in a real system. Within this definition, we explore different types of uncertainty. Section 4 and Section 5 discuss fine-grained approaches in resolving the uncertainty during context management. Section 6 demonstrates the feasibility of our model and provides a preliminary evaluation result. Section 7 compares our work with the recent research works in dealing with context uncertainty. Finally, in Section 8 we summarise our work and outline the future direction of this research.

2. SENSORS

Pervasive computing systems operate in large, open and ever-changing environments, where a huge number of sensors of different types are involved. These sensors can be categorised into types according to the type of information they provide: *environmental* sensors are those which generate information from the real world, for example noise level, temperature, humidity, etc.; *positioning* sensors that locate or track the movement of objects; *device* sensors that report the state of hardware and equipment, e.g., whether a printer is busy, idle, or off; *virtual* sensors that extract information from other software or applications (for example, a virtual sensor could be used to mine schedule information from an online calendar).

A typical approach to representing the characteristics and possible imperfections of sensed data is to describe sensor fidelity as meta-information in a quality matrix. We assume that there should exist different types of quality matrix for each category of sensor and individual sensors. Further, different types of sensors should have different quality parameters that can be applied to the data they output.

We propose a general quality matrix that can be used to describe any type of sensor. The metadata consists of *frequency*, with a list of *accuracy* and *precision* pairs. Frequency is defined as the sample rate – how often the sensor data is updated. The resolution and frequency are determined by the technical specification of sensors given by the manufacturer. Precision defines the range and accuracy is the percentage of how often the accuracy is achieved [10]. Different ranges of precision result in different accuracies. For example with our in-house location system Ubisense, to achieve 70% accuracy, the precision on x- and y-axis are 3.30 and 2.22 meters. Accuracy and precision can be acquired through different approaches for example training from experimental set up or calculated from component diagnostics.

The quality matrix of a given sensor should be referenced by all sensed data when it is produced. The general quality matrix as we describe is not definitive – it should be extended with more quality parameters for a particular type of sensor.

These sensors are the inputs that drive the production and derivation of context. This implies that the imperfections of sensed data are one of the causes of context uncertainty. There are at least three factors that are responsible for imperfect sensor data:

- *technical limitation* of sensors: each sensor is produced with inherent errors. This is due to the manufacturing process and hardware limitations. When sensors are installed, they may suffer from breakdown, disconnection from network, or signal delay;
- *environment noise*: the accuracy of some sensors may be subject to radio interference, temperature, humidity, sound noise or reflective materials which signals bounce off;
- and *users*: the configuration of sensors by users may affect the accuracy of sensors. In addition, especially for physical devices (like tag-based positioning sensors), the reliability of their data will be decreased if users do not correctly use them.

Technical limitations of sensors are reasonably fixed. They can be provided by the manufacturer or empirically calculated after installation. We can combine these values with environmental noise which is intermittently gathered through sampling and machine learning algorithms. In contrast, the influence of users on the process of gathering accurately sensed data is far more unpredictable.

When users are taken into account, the confidence of the sensor data can be computed by a function that takes the precision and the impact factor of the use. For example, in Middlewhere [20], the confidence on data from a tag-based location sensor is the product of its accuracy and the probability that a user wears a tag.

3. CONTEXT AND UNCERTAINTY

3.1 Context

Context can be categorised into different data types according to their particular properties. Each context type indicates a set of context values that represent reality entities or one property (that is, aspect) of reality entities. For example, the *Location* context type contains a set of location data that represent the location property of entities, such as a coordinate or a place with a human-friendly name [27]; the *Person* context type that contains a set of person entities, or social communities; and the *Environment* context type that contains a set of physical properties about an environment entity like temperature, humidity, or noise level.

A context type has a set of ground values, labelled as V_g, that are the irreducible (the smallest perceivable grained) elements. The *tangible context value* of a context type is defined as a set of its ground values through a mapping relationship: $m : V \rightarrow 2^{V_g}$. For example in the *Location* context type, the ground values are a set of single coordinate points, like [12.22,5.26,0.09], and a tangible context value "Lecture Room 01" maps to a set of coordinates that are in this room. In the *Temperature* context type, the ground values are a set of individual degrees in a certain unit (e.g.,

Celsius or Fahrenheit scale), like 23°C, and a tangible context value "warm" maps to a certain range of degrees that are considered as "warm".

In our model only tangible values can be managed or used, while ground values are acted as a meta-data for a context type that cannot be accessed. (All the context values mentioned in this paper are tangible values.) Tangible values vary in different granularities (or abstraction levels). The precondition of comparing the granularities of two values is their comparability. Two tangible values v_i and v_j are not comparable, if $m(v_i) \cap m(v_j) = \emptyset$. A more general partial order of granularity is out of the scope of this paper, for example "CSI building in UCD" is finer-grained than the city "Cork", but they are not comparable in our model.

Definition 1. *In a context type, a tangible value v_i is called finer-grained than v_j, labelled as $v_i \sqsubseteq v_j$ iff $m(v_i) \subseteq m(v_j)$.*

For example, in the Location context type, a place named "Lecture Room 01" is finer-grained than a place named "CSI building", since the coordinate set of "CSI building" is a super set of that of "Lecture Room 01" as shown in Figure 1 (a). As another example, consider the Temperature context type, a human-friendly word "cold" is coarser-grained than another word "freezing" since the temperatures mapping to "cold" contain those mapping to "freezing" as shown in Figure 1 (b).

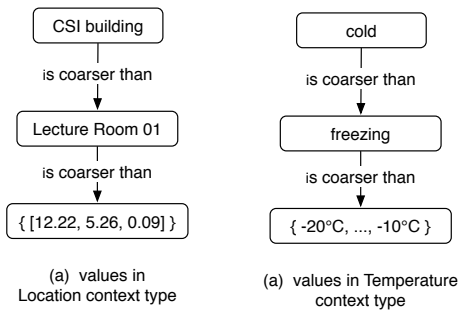

Figure 1: An example of granularities in Context Types

A number of authors [5, 23] have advocated RDF (Resource Description Framework) as a straightforward and well-founded means for modeling context. A piece of context information is modeled in a triple (s, p, o) that indicates that a subject s is associated with an object o in a relationship p. Both subject and object are tangible values of certain context types, which are a set of the ground values in corresponding context types. For example, `({erica}, hasLocation, Lecture Room 01)` indicates that a person named "erica" is located in a place named "Lecture Room 01".

There can exist multiple context predicates between values of two context types, each of which represents one semantics by linking two context values with a certain relationship. In addition, a predicate can be defined on two values in a single context type. For example, an adjacency relation *adjacent* can be defined on Location context, indicating that one symbolic place is spatially adjacent to another place.

Two features will be explored on a RDF triple: *when* and *how confident* two context values are valid in this predicate. Chaari et al. [3] provide a generic representation of context which includes a temporal characteristic and quality measure. We will use the same definition on context, which is formalised as follows.

Definition 2. *Context is represented as a tuple: $c = ((s, p, o), t, conf)$*

- *(s, p, o) is a RDF triple, whose subject and object are tangible values of two context types;*
- *t is a life span of a triple, which can be a time instant or a time interval with a starting time t_s and an ending time t_e;*
- *and $conf$ ($\in [0, 1]$) is a context confidence degree to which the subject and object are valid in this predicate.*

For example, a context is modeled as `({erica}, hasLocation, {[12.22,5.26,0.09]}),` "`2007-10-26 09:36:02`", `0.7)`, indicating that a person named "erica" was located at a coordinate point [12.22,5.26,0.09] at a time instant "2007-10-26 09:36:02" with a confidence 0.7.

For the life span of a context, its starting time is the time when this context is acquired and its ending time is the time to which this context is valid. The gap can be defined according to the sample rate of a sensor or profiled by the system developer.

The semantics of the context confidence is a system's belief in the truth of a context. When it comes to how to obtain the context confidence, we need to consider how a context is acquired. Henricksen *et al* [13] have summarised three approaches to acquire context from sensors: sensing from physical sensors (including the environmental, positioning, and device sensors discussed in Section 2), profiling from users or from application that track user inputs (e.g., virtual sensors), and inferring from other sensor data. Sensed context suffers from the imperfection of sensor data, whose confidence is the reliability degree on sensor data; profiled context suffers from the infrequent update of information, whose confidence is the reliability degree on the person for inputting information; and derived context suffers from oversimplified reasoning mechanisms, whose confidence is the belief degree on the derivation rule [13].

Data from physical sensors are transformed to be one piece of context information; for example, a positioning sensor only provides context about an entity's location. The context confidence is the sensor confidence as computed in Section 2.

Data from virtual sensors can be translated to multiple pieces of context information; for example, data from an online calendar sensor can produce the following contexts: a scheduled event and the contexts about the event including time, attendances, and location. The sensor confidence on this virtual sensor indicates the percentages of how often the scheduled event occurs. However, the confidences on each of these component contexts are different: whether this event occurs at a scheduled time in a scheduled location or all the attendees attend this event. Thus, the confidence on context from virtual sensors can be either the sensor confidence or the conditional probability on the sensor confidence (e.g., given that the scheduled event occurs, how often the event occurs in a scheduled location).

For the inferring approach, context is indirectly acquired from sensor data. For example, an activity sensor that

monitors the use of a computer's keyboard and mouse by a logged-in user by recording the time of the the last keystroke. If this computer is a desktop, then the sensor data can be used to imply the user's location – where the computer is. For example, when the time gap between two continuous keystrokes is less than 10 seconds, then the user is very likely to be in the location. When the time gap is over 1 hour, then the user is much less likely to be there during this hour. The sensor confidence indicates that how often the last keystroke is sensed from the logged-in user. The confidence on implied user's location depends on the sensor confidence and the probabilities associated with the specified rules.

In summary, context is acquired from sensors, but different acquisition processes will result in different computations of context confidence. The above discussion presents a detailed analysis of how context confidence is related to sensor confidence.

3.2 Uncertainty

Each context is subject to uncertainty to different degrees. As demonstrated by Henricksen *et al*, the uncertainty of context can be characterised as *out of date*, *incomplete*, *imprecise*, or *conflicting* [13]. In the following, we will analyse the semantics of these different uncertainties within the above context definition.

The definition of context consists of three parts: a RDF triple, a life span, and a context confidence. The RDF triple is used to evaluate the characterised uncertainty on context values, such as incompleteness, imprecision, and conflict. A context $c = ((s, p, o), t,\ conf)$ is called

- *incomplete*, if $s = \emptyset$ or $o = \emptyset$;
- *imprecise*, relative to a certain resolution r on o (or s), if $|m(o)| > |m(r)|$ (or $|m(s)| > |m(r)|$); that is, the the tangible value covers a larger set of ground values than that of the required resolution;
- or *conflicting* with another context $c' = ((s', p, o'), t',\ conf')$, only if they reflect inconsistent states of the reality world at a certain time. The precondition of the evaluation is that these contexts share the same semantics: the same predicate. The evaluation of conflict is performed by comparing their life spans and their corresponding context values. The life spans of two contexts are comparable if $t.t_s \leq t'.t_e \leq t.t_e$ or $t.t_s \leq t'.t_s \leq t.t_e$. There are three semantics of conflict. c is
  - *consistent* with c', if context values in c are finer-grained than those in c'; that is, $s \sqsubseteq s'$ and $o \sqsubseteq o'$, as shown in Figure 2 (a);
  - *partially conflicting* with c', if context values in c overlap with those in c'; that is $m(s) \cap m(s') \neq \emptyset$ and $\neg(s \sqsubseteq s')$ and $\neg(s' \sqsubseteq s)$ and $m(o) \cap m(o') \neq \emptyset$ and $\neg(o \sqsubseteq o')$ and $\neg(o' \sqsubseteq o)$, as shown in Figure 2 (b);
  - or *completely conflicting* with c', otherwise; that is, $m(s) \cap m(s') = \emptyset$ or $m(o) \cap m(o') = \emptyset$, as shown in Figure 2 (c).

The life span can be used to evaluated whether a context $((s, p, o), t,\ conf)$ is out of date relative to a certain time t_c and a time gap τ. If $t < t_c - \tau$ or $t.t_e < t_c - \tau$, then this context is considered *out of date*.

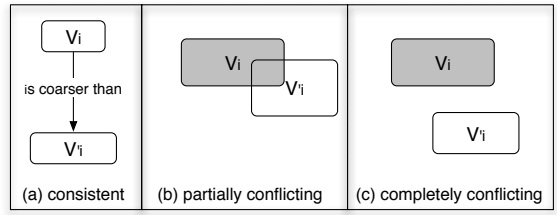

Figure 2: Semantics of conflicting contexts, where v_i and v_i' are tangible values in a certain context type

4. CONTEXT INTEGRATION

In pervasive computing environments, different sensors may provide context with the same predicate. These sensor data vary in granularity of values, timeliness (how often sensors produce data), or reliability (how reliably these sensor produce correct data). This huge number of potentially conflicting and redundant contexts will make a system awkward and inefficient, especially when a system attempts to provide a responsive service. It is expected to immediately integrate sensor data into a small number of accurate contexts. In the following, we will discuss a general approach to integrate contexts under the consideration of two factors: context values and life span of each context.

Assume that there have been a number of available contexts in this system: $c_i = (s, p, o_i), t_i, conf_i)$ $(0 \leq i \leq n)$, which share the same predicate p, and have different life spans and confidence values. These multiple pieces of context information exhibit different states of reality at a certain time, which share the same semantics (predicate) and the same focus (subject). Context integration is used to combine them in order to *determine the most accurate state for the current reality*. Hence, a context integration is expressed as a request: $((s, p, ?), t, ?)$ to decide the object value, given the same focus s, the same predicate p, and the required time t (e.g., that can be defined by developers; or required by applications). The question is to determine the most accurate value for this context relation within the required time. The procedure will be carried out in three steps: *computing the time factor*, *computing the value factor*, and *combining these two factors*.

4.1 Time Factor

Our assumption is that the more recent a sensor datum (i.e. whose life span is closer to the required time), the more reliable it is. To compute the time factor, we extend the required time t to be $[t-\tau, t]$, where τ is a threshold value, if t is a time instant. The selection of τ is application- and sensor-specific. Sensor data will be considered only when their life span are within this range. For each available sensor datum, the freshness of its life span is evaluated according to the linear relation of time:

$$freshness(t_i, t) = \mathbf{max}(1 - \frac{t - t_i.t_s}{\tau}, 0), \qquad (1)$$

where $t_i.t_s$ is the time from which a given context is valid.

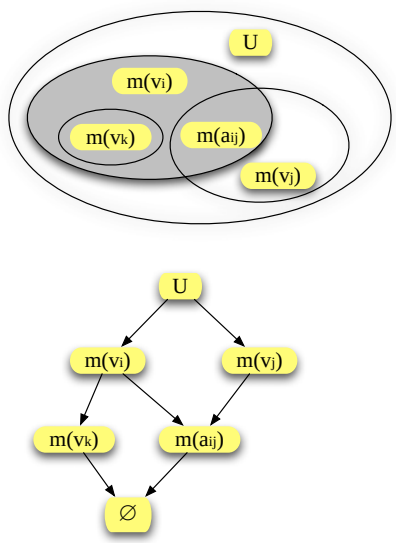

Figure 3: Different context values interact with each other

4.2 Value Factor

To compute the value factor, we will consider how different context values interact with each other with the relationships discussed in Section 3.2. The *conditional confidence* on a context value is the confidence of how a system believes this value given all the related context values. We assume that if one context value interacts with another (that is, these two values are consistent or overlapping), then their conditional confidence given all the interacted context values will be affected (increased or decreased). This updated conditional confidence is caused by the overlapped value that is supported by multiple context values. Figure 3 shows how a context value v_i interacts with another two context values v_k and v_j: v_i is coarser-grained than v_k and is overlapping with v_j on a_{ij}. The conditional confidence on v_i given v_j and v_k should be no less than the conditional confidence when v_i had not interacted with them. The conditional confidence on v_i will be revised, considering the conditional confidence on a_{ij} given both v_i and v_j, and that on v_k given both v_i and v_k.

Bayes' Theorem will be used to compute the conditional confidences on each context value whose semantics is how probable the context value is true, given all the other available context values. The computation of conditional confidence on each given context value involves the following procedures:

- choosing the *optimal confidence* $conf^p$ on every overlapped value (using Equation 2);
- calculating the *interacted conditional confidence* $conf^c$ on each given context value using the interacted confidence on overlapped values (using Equation 3);
- and calculating the *non-interacted conditional confidence* $conf^s$ on each context value if it does not interact with any other value (using Equation 4).

Equation 2 will give a formula to choose the optimal confidence $conf^p$ on a overlapped context value that is supported by several given context values. Its semantics is a system's belief in the truth of a value a_{ij}. If a_{ij} is overlapped by two context values v_i and v_j, then the belief on a_{ij} is determined by the maximal belief in the truth of either v_i or v_j:

$$conf_{ij}^{p} = \mathbf{max}(conf_i, conf_j) \tag{2}$$

The interacted conditional confidence $conf^c$ on v_i is the conditional confidence given all the interacted values in v_i using Equation 3.

$$\begin{aligned} conf_i^{c} &= Pr(v_i|a_{i1},\ldots,a_{in}) \\ &= \frac{Pr(a_{i1},\ldots,a_{in}|v_i)Pr(v_i)}{Pr(a_{i1},\ldots,a_{in}|v_i)Pr(v_i) + Pr(a_{i1},\ldots,a_{in}|\neg v_i)Pr(\neg v_i)} \end{aligned} \tag{3}$$

$$\begin{aligned} Pr(a_{i1},\ldots,a_{in}|v_i) &= \prod_{j=1}^{n} Pr(a_{ij}|v_i) \times Pr(v_i); \\ Pr(a_{ij}|v_i) &= conf_{ij}^{c} \times m(a_{ij})/m(v_i) \\ &\quad +(1 - conf_{ij}^{c})(1 - m(a_{ij}/m(v_i))); \\ Pr(a_{i1},\ldots,a_{in}|\neg v_i) &= \prod_{j=1}^{n} Pr(a_{ij}|\neg v_i) \times Pr(\neg v_i); \\ Pr(a_{ij}|\neg v_i) &= 1 - conf_{ij}^{c}; \\ Pr(v_i) &= m(v_i); \\ Pr(\neg v_i) &= 1 - m(v_i); \end{aligned}$$

where, U is the universal set of the ground value of C, $m(v_i)$ is the function that maps v_i to a subset of U, and $conf_{ij}^{c}$ is the conditional confidence on the overlapped value a_{ij} if it also interacts with finer-grained context values.

Given any context value v_i in a context type C, if it is not supported by any other context value (that is, no value is finer-grained than it or overlapping it), its non-interacted conditional confidence $conf_i^{s}$ is computed as:

$$conf_i^{s} = \frac{conf_i \times m(v_i)/U}{conf_i \times m(v_i)/U + (1 - conf_i)(1 - m(v_i)/U)} \tag{4}$$

where $conf_i$ is the context confidence on the value v_i.

Equation 4 shows that a system's belief in the truth of a context value given all the possible values for this context type. $conf_i \times m(v_i)/U$ is how possible v_i is chosen from the universal set when v_i is true state of reality, and $(1 - conf_i)(1 - m(v_i)/U)$ is how possible v_i is chosen when v_i is not true as shown in Figure 4.

For example in Figure 3, it is assumed that initial confidences $conf_i$, $conf_j$, and $conf_k$ on context values v_i, v_j, and v_k are 0.6, 0.8, and 0.7 respectively, and $m(v_i)/U$ is 0.2, $m(v_j)/U$ is 0.2, $m(v_k)/m(v_i)$ is 0.4, and $m(a_{ij})/m(v_i)$ is 0.3. Since a_{ij} is supported by both v_i and v_j, its optimal confidence is $conf_{ij}^{p} = \mathbf{max}(0.6, 0.8) = 0.8$. The conditional confidences on v_i and v_j are 0.686 and 0.5 respectively. The non-interacted confidence on v_k is 0.169. This shows that when a context value interacts with multiple context values, it is more likely to be selected as the most accurate answer even if its initial confidence is not highest. It is implied that more supported evidence (the confidence on overlapped

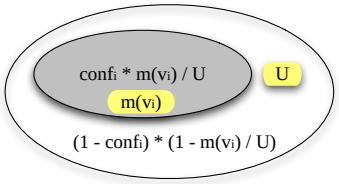

Figure 4: The conditional confidence on a single context value in Equation 4

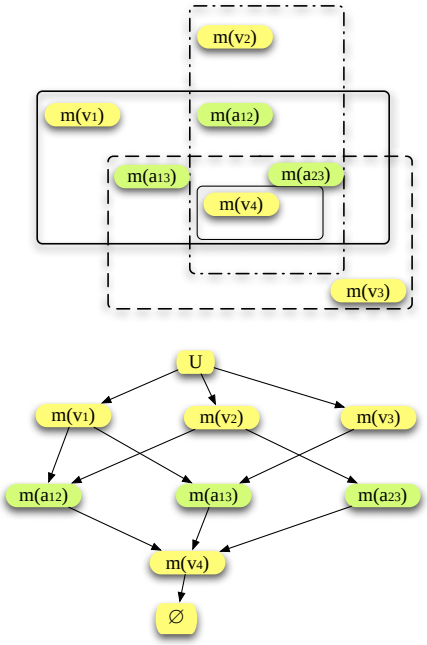

Figure 5: Multiple context values are involved in a more complicated interaction

value or that on the contained value is greater than 0.5) will increase the belief on the truth of a context value.

The above section describes the atomic equations on how to compute the conditional confidence on each given context value when they interact with each other. In a real system, multiple context values can be involved in a complicated interaction as shown in Figure 5. We need a structure to organise these values and examine their value relationships like the lower part of Figure 5.

So far, all the discussed value relationships are based on a set of ground values that are mapped to each tangible value. To deal with a complicated interaction, we organise these context values into a partially ordered set L, whose partial order is the granularity $\sqsubseteq$ defined in Definition 1. L consists of a set of given context values and their overlapped values. These overlapped values demonstrate how these given context values interact with each other.

Each value e records a set S^c of the immediately coarser-grained values in L and a set S^f of the immediately finer-grained values. These two sets are satisfied by Lemma 3.

Lemma 3. *For any $e_i \in S^c \subseteq L$, $e \sqsubseteq e_i$ and there does not exist a value $e_j (e_j \neq e_i)$ in L such that $e \sqsubseteq e_j \sqsubseteq e_i$.*

For any $e_i \in S^f \subseteq L$, $e_i \sqsubseteq e$ and there does not exist a value $e_j (e_j \neq e_i)$ in L such that $e_i \sqsubseteq e_j \sqsubseteq e$.

When this partially ordered hierarchy is constructed, we will revise their confidences using Algorithm 1.

Algorithm 1 Compute Conditional Confidences

INPUT: $L = \{(e_i, conf_i) | e_j \sqsubseteq e_i, 1 \leq i < j \leq n\}$
OUTPUT: $L = \{(e_i, conf_i^c) | e_j \sqsubseteq e_i, 1 \leq i < j \leq n\}$
// compute the conditional confidence from the coarsest-grained values to the finest-grained values
for i from 1 to n **do**
  // start from the coarsest-grained value
  if e_i is an overlapped value and S_i^c is not empty **then**
    $conf_i^p = maxConf(S_i^c)$
    // compute the interacted conditional confidence on e_i given all the coarser-grained values, using Equation 2
  end if
end for
// update the conditional confidence from the finest-grained values to the coarsest-grained values
for i from n to 1 **do**
  // start from the finest-grained value
  if S_i^f is empty **then**
    $conf_i^s = noninteractConf(e_i, conf_i)$
    // update the confidence on e using Equation 4
  else
    $conf_i^c = interactConf(S_i^f)$
    // update the confidence on e_i using Equation 3
  end if
end for

Whenever a context integration is required, a system will collect all the available context values from sensors and then organise these values into a partially ordered set. From the coarsest-grained values to the finest-grained ones, every overlapped value will get the optimal confidence from its coarser-grained values using Equation 2. After this process, we start another process to compute the conditional confidence on each interacted context value using Equation 3 from the finest-grained values to the coarsest-grained ones. For a context value that does not involve in any interaction, its conditional confidence will be calculated using Equation 4. To determine the most accurate value among all the given context values, we will combine their conditional confidences and the evaluation on their freshness: $conf_i^c \times freshness(t_i, t)$. The value with the highest score is the result.

5. CONTEXT ABSTRACTION

To support more specific pervasive services, a system should be able to identify more complicated and higher-level context, called *situation*. A situation abstracts the invariant characteristics of contexts and their combination [24]. Once the current contexts meet the characteristics, a situation is considered *occurring*. The invariant characteristics can be interpreted as constraints on values and life span of contexts:

- *Constraints on a context value* confine a context value to a sub set of ground values for this context type;

- *Constraints on a context life span* requires that the context must be valid in a certain time, or it must meet a certain temporal relationship.

Constraints are expressed in a *specification* of a situation. Loke [17] represents situations as a set of sensor constraints connected by logical operatives. Instead of specifying situations in terms of particular sensor constraints, we define a more general specifications in terms of constraints on context $l : E(r_1, \ldots, r_n)$, where r_i is the above constraints on a certain context predicate, and E applies the logical connectives (AND, OR , and existential and universal quantifiers) on them.

Example 4. *A specification of a meeting situation consists of the constraint on the calendar – a meeting scheduled; a person's location – in a scheduled meeting area; and the constraint on the current time – in the scheduled meeting time. This can be represented as follows:*

$$(c, scheduled, meeting)$$
$$\wedge\, (t_{current}, \mathbf{isIn}, t_{scheduledMeetingTime})$$
$$\wedge\, (l_{current}, \mathbf{isContainedIn}, l_{scheduledMeetingLocation}).$$

To identify a situation, we will take a set of contexts with these contexts, and evaluate whether they satisfy the corresponding constraints. The evaluation procedure will be taken in two steps: evaluating each constraint r_i and combining them according to their logical description in a specification.

To evaluate c_i, we consider two factors: the confidence $conf_i$ associated with a context, and the match degree mat_i to which this context satisfies the constraint. The match degree will be evaluated in the two types of constraints: constraints on a context life span and constraints on values.

The constraint on the occurrence time of a context relation is manifested in two different ways: by requiring the context to occur in a certain time interval or by specifying a certain temporal sequence. At our current research stage, we only consider the former circumstance. The match degree mat^t on the occurrence time constraint is computed as:

$$mat^t = \begin{cases} 1.0, & t.t_s \geq t_r.t_s \text{ and } t.t_e \leq t_r.t_e; \\ 0.0, & t.t_s > t_r.t_e \text{ or } t.t_e < t_r.t_s; \\ \frac{t.t_e - t_r.t_s}{t.t_e - t.t_s}, & \text{otherwise.} \end{cases} \quad (5)$$

where t is a life span of a context, and t_r is required time interval in a constraint. If t is a time instance, it will be converted to a time interval: [t - ϵ, $t + \epsilon$], where ϵ is set by developers according to different contexts and application requirements.

The constraints on the values of a context can be different with respect to the semantics of its predicates and the properties of values. Thus, there does not exist a uniform formula to compute all the value match degrees mat^v. Instead, we focus on the quantification constraints. A universal quantification in a value constraint is required that any value v in a set of contexts (v, p, o) (or (s, p, v)) should satisfy a constraint r_v. The match degree for this universal quantification can be intuitively computed in the framework of fuzzy set:

$$mat^v = \mathbf{min}(conf_1 \times \gamma_1, \ldots, conf_n \times \gamma_n),$$

where $conf_i$ is the confidence on each (v, p, o) (or (s, p, v)), and γ_i is the degree to which a value v matches the constraint c_v.

An existential quantification in a value constraint requires that there exists at least one value v in a set of contexts (v, p, o) (or (s, p, v)) that satisfies a constraint r_v. The match degree for this existential quantification can be computed as:

$$mat^v = \mathbf{max}(conf_1 \times \gamma_1, \ldots, conf_n \times \gamma_n),$$

The final match degree mat on a context constraint is computed by combining the match degree on the life span and that on the value constraint:

$$mat = \mathbf{min}(mat^t, mat^v).$$

To determine a degree to which a situation's specification is satisfied, we will consider the logical connectives (AND, OR, NOT) that are applied on individual constraints:

If $r_i \wedge r_j$, then $mat(r_i \wedge r_j) = \mathbf{min}(mat_i, mat_j)$;
If $r_i \vee r_j$, then $mat(r_i \vee r_j) = \mathbf{max}(mat_i, mat_j)$;
If $\neg r_i$, then $mat(\neg r_i) = 1 - mat_i$.

where mat_i (or mat_j) is a match degree on a constraint r_i (or r_j).

6. DEMONSTRATION AND EVALUATION

This section demonstrates the feasibility of our model and provides a preliminary evaluation result. We gathered two simple datasets [1] in our office environment as a proof of concept. There are four sensors producing context: Ubisense, Bluetooth, Activity and Calender sensors. We use these datasets as training data to provide sensor quality values.

Ubisense is a tag-based positioning sensor network, which is used to track an object's real-time location in an indoor environment. Ubisense provides a precise location of a person in the form of coordinates. The coordinates can be mapped to a place with a human-understandable name. Ubisense data suffers from all the uncertainty factors described in Section 2. We have analysed the imperfectness of Ubisense data [6] in the dataset, which has three main causes: the environment where the devices are installed, a tag's state (when a tag enters an idle state after a period of inactivity, it will stop producing data), and the use of a tag (sometimes users forget to carry their tag with them; however, in our dataset, the user is aware of being involved in an experiment, so she always carried the tag). Each Ubisense reading is represented in a location context as a rectangle [27] whose center is the Ubisense reading coordinate and whose width and length is the x- and y-axis precision: 3.30m and 2.22m. The confidence on this context is the corresponding accuracy of Ubisense 70%, as determined by training our dataset.

Bluetooth sensors constantly scan for mobile Bluetooth devices (like a mobile phone or a laptop). They record the IP address of the Bluetooth server, the MAC address of the device, the signal strength, and the timestamp [15]. We convert the Bluetooth sensor data into a location context as a rectangle whose center is the coordinate of the Bluetooth center and whose width is the precision corresponding to

[1]The sample data are published online here: `http://kind.ucd.ie/~juanye/datasets/Ria2008Dataset.zip`.

Frequency	≤10	≤30	≤60	≤300	≤600	>600
Confidence	82.4%	11.9%	4.0%	1.3%	0.3%	0.1%

Table 1: Confidence on inferred location with different frequencies of sending the "active" state

the current signal strength (for example, when the signal strength is strong, the precision is set as 1 meter and the accuracy is 57.7%). The accuracy of Bluetooth sensor data is low since we use a naive approach to training and representing Bluetooth data. More advanced techniques will be used in the future, such as those identified by Bell et al. [2].

Activity sensors sense the activities of keyboard and mouse. If the keyboard or mouse is used, then this sensor will send an "active" state to indicate that the computer is being used at the moment. The frequency that it sends the data indicates the usage status of the computer. The confidence on the activity data is 1.0, since the computer is always used by the logged-in user in the dataset. When the computer is a desktop and it is logged in by a registered user, then we use their data to infer the user's location. Table 1 shows that the confidences of the user being located in the place of the computer with the different frequency of sending the "active" state, as determined by our training dataset.

Calendar sensors capture the scheduled events in the personal and group calendars. An event can be coarse-grained, for example, a person is on holiday during a period; while it can also be fine-grained, for example, a meeting event is scheduled, by specifying its attendees, starting time, end time, location, and content. In the dataset, all the scheduled events occurred, so the confidence on the events is 1.0.

To evaluate the context integration approach, we will gather all these sensor data to decide a user's location (the finest-grained human-friendly place, like a desk area) at a certain time instant. We gathered the sensor data for certain places over a period. The integrated location context is more accurate than any of individual sensor data. Since Ubisense and Bluetooth location data are represented in a rectangle that consists of coordinates, they need to be mapped to a human-friendly place that the rectangle takes the maximum coverage ratio. For example a user is working in her office from "18-04-2008T14:40:00" to "18-04-2008T14:52:30", the accuracy of identifying a user's location in an office is 88.5% using Bluetooth and Ubisense data. When a user starts using her desktop computer, the activity sensor produces the implied location: the user's desk area. This sensor data not only gives support to Bluetooth and Ubisense data to various degrees, but also it identifies more precise location with higher accuracies 95%.

The complexity of the context integration process is $O(2^n)$, where n is the number of input context values and new added values, which is caused by the procedure of organising different location context values into a partially ordered set. However in a realistic environment, the number of the context values that are to be integrated at a certain time instant will not be greater than five (e.g., in our experiment, the location data will be at most three).

We evaluate the context abstraction technique using Example 4: to determine whether a person named "erica" is in a meeting situation at a given time "23-10-2007T11:5:00Z". The available contexts include:

- `(({erica}, hasLocation, [12.22,5.26,0.09], [15.52, 7.48, 0.09]), ["2007-10-23 11:14:59," "2007-10-23 11:15:00"], 0.7)`, indicating that Ubisense located erica in a rectangle area with two coordinates, whose confidence is 0.7;
- `(({erica's calendar}, scheduled, meeting)`, "2007 -10-23 09:00:00", 1.0), indicating that erica's calendar scheduled a meeting at a time "2007-10-23 09:00: 00" with a confidence 1.0;
- `((meeting, scheduledTime, ["2007-10-23 11:00:00", "2007-10-23 11:30:00"]), "2007-10-23 09:00:00", 0.65)`, indicating that the scheduled meeting time with a confidence 0.65 – how possible the meeting occurs during this scheduled time;
- `((meeting, scheduledLocation, coffee area), "2007 -10-23 09:00:00", 0.75)`, indicating that the scheduled meeting location with a confidence 0.75 – how possible the meeting occurs there.

The match degree on time is 1.0, using Equation 5. The match degree on the location value is $0.7 \times 0.9 = 0.63$, where 90% of the rectangle area overlaps with the coffee area. Thus the confidence on the occurrence of a meeting situation is 0.63.

7. RELATED WORK

Henricksen et al. [13] explored context information quality for sensed, profiled and derived context. Their context modeling language (CML) allowed facts to be associated with relevant quality indicators that allow the end users of the information to make judgements about the level of confidence they invest in it. Each fact was associated with zero or more quality parameters. Similarly, Gray et al. [11] introduced two quality matrices for sensor data and sensors respectively. The quality matrix for sensor data includes coverage, resolution, accuracy, repeatability, and frequency. The quality matrix for sensors includes the reliability, intrusiveness, and security or privacy. We also suggest using separate quality matrices on the type of sensor and data from an individual sensor in this type. The former quality matrix can be referenced by all the data sensed from this type of sensor, while the latter matrix reflects the quality of data from a particular sensor, which is affected by the factors mentioned in Section 2.

This quality approach is good at expressing uncertainty. It is qualitative so it lacks a formal uniform technique of managing the uncertainty in a quantitative way, even though Lei et al. [16] and Cohen et al. [4] proposed an approach to aggregate the quality matrix on different contexts. Using this approach, context models either provide an application (or sensor)-specific mechanism to integrate context according to the quality data, or simply pass along the quality data with context information to applications. The quality approach also fails to capture the uncertainty during the process of deriving new context information.

In contrast, the quantity approach allows context to be associated with a confidence value that can be a fuzzy value or a probability value [19]. Different techniques (like fuzzy logic, Bayesian networks, or the Dempster-Shaffer theory) are applied to deal with the uncertainty with respect to different semantic interpretations on confidence values.

Anagnostopoulos et al. [1] defined a fuzzy function to evaluate the degree of membership in a situational involvement that referred to the degree of belief that a user was involved in a predicted situation. They defined Fuzzy Inference Rules (FIR) that were used to deal with imprecise knowledge about situational context and the user behaviour/reaction and historical context. We also use the fuzzy function when abstracting multiple lower-level contexts into a situation. Our fuzzy function is used to evaluate how much the current context satisfies the constraints in a situation's specification.

Bayesian networks have a causal semantics that encode the strength of causal relationships with probabilities between lower- and higher-level. Bayesian networks have been applied by Ranganathan et al. [19], Gu et al. [12], Ding et al. [9], Truong et al. [22], Dargie et al. [7], and Ye et al. [28]. For example, Gu et al. encoded probabilistic information in ontologies, converted the ontological model into a Bayesian network, and inferred higher-level contexts from the Bayesian network. Their work aimed to solve the uncertainty that is caused by the limit of sensing technologies and inaccuracy of the derivation mechanisms. Bayesian networks are best suited to applications where there is no need to represent ignorance and prior probabilities are available [14]. In a pervasive computing system, these probabilities are acquired either by training a number of data or from the domain experts. As a complementary technique, our approach can be applied in a system where developers have less knowledge about the environment or where training data are not sufficient.

The Dempster-Shafer theory is a mathematical theory of evidence, which propagates the uncertainties and consequently provides an indication of the quality of inferences. It has been applied to fuse multiple sensor data by Wu et al. [25] and Padovitz et al. [18]. For example, Padovitz et al. also use the concept of a situation space consisting of a set of context constraints. Based on the Dempster-Shaffer theory, they use a utility based fusion approach to fuse sensor readings that are intuitively considered to be pertinent in reasoning about context. About sensor fusion, Ranganathan et al. [20] use probabilistic reasoning to interpret conflicting and overlapping location readings from multiple sources. They use Bayes' Theorem to calculate conditional probabilities for a person's location given a set of location readings from multiple sources. They also explain that data freshness will impact context confidence and describe both a data "time to live" parameter and freshness degradation functions. However, they do not include freshness degradation in their confidence calculations. We take a similar approach to integrate values for general context types and determines the confidence on a context by considering both the original confidence and freshness of all the available contexts.

8. CONCLUSION AND FUTURE WORK

This paper provides an applicable and fine-grained approach to resolve the uncertainty of context in a pervasive computing system. We have established a simple experiment to evaluate our approach against the real dataset that was gathered in our environment. In the context integration, our underlying assumption is that the context value with the highest confidence will support other values when they interact with each other (that is, overlapping or containing). We use Bayes' Theorem to calculate the conditional confidence on each involved value given all the values. The preliminary result shows that the accuracy of integrated context values is higher than any individual sensor data. However, the dataset we gathered is limited to certain places where both Ubisense and Bluetooth sensors can get the best signal. In addition, we use a naive technique to train and represent the sensor data, which decreases the performance of sensors. In the context abstraction, we evaluated the propagation of uncertainty in a meeting scenario. We have not considered complicated temporal constraints. In the future, we will gather a larger dataset that covers more areas and include more activities. We expect that more noise will be introduced, which will complicate the integration and abstraction approaches. We will apply more intelligent techniques (e.g., transferable belief model [21]) to resolve the uncertainty.

9. REFERENCES

[1] C. B. Anagnostopoulos, Y. Ntarladimas, and S. Hadjiefthymiades. Situational computing: An innovative architecture with imprecise reasoning. *Journal of Systems and Software*, 80(12):1993–2014, 2007.

[2] David Ferrández Bell, Juan-Carlos Cano, and Pietro Manzoni. Evaluating Bluetooth performance as the support for context-aware applications. In *ICCCN 2003: Proceedings of the 12th International Conference on Computer Communications and Networks* , pages 345–350, 20-22 Oct. 2003.

[3] Tarak Chaari, Dejene Ejigu, Frédérique Laforest, and Vasile-Marian Scuturici. A comprehensive approach to model and use context for adapting applications in pervasive environments. *Journal of Systems and Software*, 80(12):1973–1992, 2007.

[4] Norman H. Cohen, Hui Lei, Paul Castro, John S. Davis II, and Apratim Purakayastha. Composing pervasive data using iQL. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 94, Washington, DC, USA, 2002. IEEE Computer Society.

[5] Lorcan Coyle, Steve Neely, Gaëtan Rey, Graeme Stevenson, Mark Sullivan, Simon Dobson, and Paddy Nixon. Sensor fusion-based middleware for assisted living. In *ICOST'2006: Proceedings of the first International Conference On Smart homes & heath Telematics: "Smart Homes and Beyond"*, pages 281–288. IOS Press, 2006.

[6] Lorcan Coyle, Juan Ye, Emerson Loureiro, Stephen Knox, Simon Dobson, and Paddy Nixon. A proposed approach to evaluate the accuracy of tag-based location systems. In *Ubicomp 2007 Workshops Proceedings of the First Workshop on Ubiquitous Systems Evaluation*, pages 292 – 296, Innsbruck Austria, September 2007.

[7] Waltenegus Dargie. The role of probabilistic schemes in multisensor context-awareness. In *Proceedings of Fifth Annual IEEE International Conference on the PerCom Workshops '07*, March 2007.

[8] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[9] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language OWL. In *HICSS '04: Proceedings*

of the 37th Annual Hawaii International Conference on System Sciences - Track 4, page 40111.1, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Simon Dobson, Lorcan Coyle, and Paddy Nixon. Hybridising events and knowledge as a basis for building autonomic systems. *IEEE TCAAS Letters*, 2007. To appear.

[11] Philip Gray and Daniel Salber. Modelling and using sensed context information in the design of interactive applications. *Lecture Notes in Computer Science*, 2254:317–325, 2001.

[12] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A Bayesian approach for dealing with uncertain contexts. In *Proceedings of the 2nd International Conference on Pervasive Computing*. Austrian Computer Society, April 2004.

[13] Karen Henricksen and Jadwiga Indulska. Modelling and using imperfect context information. In *Proceedings of PERCOM '04 Workshops*, pages 33 – 37, Washington, DC, USA, 2004. IEEE Computer Society.

[14] James C. Hoffman and Robin R. Murphy. Comparison of Bayesian and Dempster-Shafer theory for sensing: a practitioner's approach. In *Proceedings of Neural and Stochastic Methods in Image and Signal Processing II, Vol. 2032*, pages 266–279, October 1993.

[15] Stephen Knox, Adrian K. Clear, Ross Shannon, Lorcan Coyle, Simon Dobson, Aaron Quigley, and Paddy Nixon. Towards scatterbox: a context-aware message forwarding platform. In *MRC 2007: Fourth International Workshop on Modeling and Reasoning in Context*, pages 13–24, August 2007.

[16] Hui Lei, Daby M. Sow, II John S. Davis, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *SIGMOBILE Mob. Computing Communication Review*, 6(4):45–55, 2002.

[17] Seng Wai Loke. Logic programming for context-aware pervasive computing: Language support, characterizing situations, and integration with the web. In *WI 2004: Proceedings of 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 44–50, Beijing, China, September 2004. IEEE Computer Society.

[18] Amir Padovitz, Seng W. Loke, Arkady Zaslavsky, Claudio Bartolini, and Bernard Burg. An approach to data fusion for context-awareness. In *Proceedings of the 5th International Conference on Modeling and Using Context: LNAI 3554*, pages 353–367, Paris, July 2005. Springer Berlin / Heidelberg.

[19] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 03(2):62–70, 2004.

[20] Anand Ranganathan, Jalal Al-Muhtadi, Shiva Chetan, Roy Campbell, and M. Dennis Mickunas. Middlewhere: a middleware for location awareness in ubiquitous computing applications. In *Proceedings of Middleware '04*, pages 397–416, New York, USA, 2004.

[21] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial Intelligence*, 66(2):191–234, 1994.

[22] Binh An Truong, Young-Koo Lee, and Sung-Young Lee. Modeling uncertainty in context-aware computing. In *ICIS '05: Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science*, pages 676–681, Washington, DC, USA, 2005. IEEE Computer Society.

[23] Xiaohang Wang, Jin Song Dong, Chung Yau Chin, Sanka Ravipriya Hettiarachchi, and Daqing Zhang. Semantic Space: an infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, July–September 2004.

[24] Norbert Weisenberg, Rüdiger Gartmann, and Agnès Voisard. An ontology-based approach to personalized situation-aware mobile service supply. *Geoinformatica*, 10(1):55–90, 2006.

[25] Huadong Wu, Mel Siegel, Rainer Stiefelhagen, and Jie Yang. Sensor fusion using dempster-shafer theory. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference, Anchorage, AK, USA, May 21-23, 2002*, May 2002.

[26] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 22(04):315–347, 2007.

[27] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. A unified semantics space model. In Jeffrey Hightower, Bernt Schiele, and Thomas Strang, editors, *Location- and Context-Awareness*, volume 4718 of *LNCS*, pages 103–120. Springer, 2007.

[28] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Using situation lattices to model and reason about context. In *Proceedings of MRC 2007 (coexist with CONTEXT'07)*, pages 1–12, Roskilde, Denmark, August 2007.

[29] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Representing and manipulating situation hierarchies using situation lattices. *Revue d'Intelligence Artificielle*, 2008. To appear.