Context obfuscation for privacy via ontological descriptions*

Ryan Wishart[1], Karen Henricksen[2] and Jadwiga Indulska[1]

[1] School of Information Technology and Electrical Engineering,
The University of Queensland
{wishart, jaga}@itee.uq.edu.au
[2] CRC for Enterprise Distributed Systems Technology
{kmh}@dstc.edu.au

Abstract. Context information is used by pervasive networking and context-aware programs to adapt intelligently to different environments and user tasks. As the context information is potentially sensitive, it is often necessary to provide privacy protection mechanisms for users. These mechanisms are intended to prevent breaches of user privacy through unauthorised context disclosure. To be effective, such mechanisms should not only support user specified context disclosure rules, but also the disclosure of context at different granularities. In this paper we describe a new obfuscation mechanism that can adjust the granularity of different types of context information to meet disclosure requirements stated by the owner of the context information. These requirements are specified using a preference model we developed previously and have since extended to provide granularity control. The obfuscation process is supported by our novel use of ontological descriptions that capture the granularity relationship between instances of an object type.

1 Introduction

The use of context information has been widely explored in the fields of context-aware applications and pervasive computing to enable the system to react dynamically, and intelligently, to changes in the environment and user tasks. The context information is obtained from networks of hardware and software sensors, user profile information, device profile information, as well as derived from context information already in the system. Once collected, the context information is usually managed by a context management system (CMS) in a context knowledge base as a collection of context facts. The pervasiveness of the context collection means that the context facts provide ample fodder for malicious

* The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Education, Science, and Training).

entities to stage attacks on the context owner. These attacks may range from annoying targetted advertisements to life threatening stalking. As the information was collected to be used by the pervasive network and context-aware applications, total prevention of context information disclosure is counter productive. Thus, privacy mechanisms are needed to protect the context owner's privacy by applying access control to queries on the context information. These privacy mechanisms either authorise or prohibit context information disclosure based on a set of preferences established by the context owner.

Preferences that only allow or deny access to the context information are very course-grained, and prevent context owners from releasing less detailed, though still correct, context information. In this paper we present a preference mechanism that gives context owners fine-grained control over the use and disclosure of their context information. This would enable a context owner to provide detailed location information to family members, and low granularity information, e.g., the current city they are located in, to everyone else.

The mechanism we present to support this operates over different context types, and provides multiple levels of granularity for disclosed context information. The obfuscation procedure it uses is supported by detailed ontological descriptions that express the granularity of object type instances. Owner preferences for disclosure and obfuscation are expressed using an extended form of a preference model we previously developed for context-aware applications [7].

The remainder of the paper is structured as follows. Section 2 provides an overview of related work in obfuscation of context information. In Section 3 we present an example context model which we use as a vehicle for later examples. The paper then continues, in Section 4, with ontologies of object instances and our novel approach to using them to capture granularity levels for the purpose of context obfuscation. Our preference language for controlling obfuscation and expressing privacy constraints is discussed in Section 5, while the evaluation of these preferences is discussed in Section 6. An example demonstrating the operation of the obfuscation mechanism is then presented in Section 7. Finally, our concluding remarks and discussion of future work are provided in Section 8.

2 Related Work

Granular control over the release of context information was established by Hong and Landay [10] as a desirable feature for privacy protection. The notion of granular control is explored by Johnson et al. [11] who refer to it as "blurring". However, the technique is only mentioned with respect to location and spatial context administered by a Location Management System presented in the paper. Restricting the "blurring" functionality to a limited subset of the context information handled by the CMS is undesirable, as the CMS supports many types of context information, all of which should be potentially available at different levels of obfuscation.

The P3P [4] (Platform for Privacy Preferences) standard is intended for specifying information usage policies for online interactions. The supporting APPEL [3] language can be used to express user privacy preferences regarding the P3P

usage policies. Although P3P can provide data at different levels of granularity, it does not engage in obfuscation in the sense that specific data is modified to meet disclosure limits on granularity. In addition, as the preference rules either evaluate to true or false, requests on user context are either permitted or rejected. There is no concept of plausible deniability, where the system can dynamically provide "white lies" for the user or give a vague answer such as "unknown", with the interpretation left to the recipient.

The obfuscation of context information to different levels of precision is mentioned by Lederer et al. [12], who propose four different levels of context information: precise; approximate; vague; and undisclosed. This approach of using only a set number of levels hampers the user's control over the disclosure of the context. For example, assume exact location, the building the user is currently in, the suburb, and the city in which the user is located are provided as the four levels of obfuscation. A user not wanting to provide her exact location, but wanting to give a more specific value for her location than the building, such as a floor number, cannot do so. A better approach would be to support an arbitrary number of levels, with the number of levels set according to the type of context information.

A different approach is pursued by Chen et al., who discuss in [2] the development of a privacy enforcement system for the EasyMeeting project. Obfuscation of location information is supported as part of the privacy mechanism. The obfuscation relies on a predefined location ontology, which defines how different locations link together spatially. Obfuscation of other context types is not supported in their system.

A privacy preference language is presented by Gandon and Sadeh [5] that can express release constraints on context information as well as provide obfuscation of context. Obfuscation requirements are expressed as part of privacy preferences. This means that a commonly used obfuscation has to be respecified on each privacy preference. Furthermore, if obfuscation is used, the value to which the context should be obfuscated is required. As a result, this makes the obfuscation static. If a user specified that her location should be obfuscated to the city in which she is currently located, the name of that city would be specified as the obfuscation value. If she then moved to another city, the preference would become invalid. A more flexible technique is needed that looks for the current city, and uses that value rather than using a hard-coded value in the preference.

In summary, an obfuscation mechanism should be applicable to a wide range of different context types, not just a single type, like location. In addition to this, the mechanism should provide an arbitrary number of levels of obfuscation. The exact number of levels should depend on the type of context information being obfuscated. The mechanism should also be able to overcome missing context information in the context knowledge base. If a value required for obfuscation is unavailable, the system should choose the next most general value that still meets the limitations specified by the context owner. Furthermore, granularity preferences that are applicable to multiple privacy preferences should only

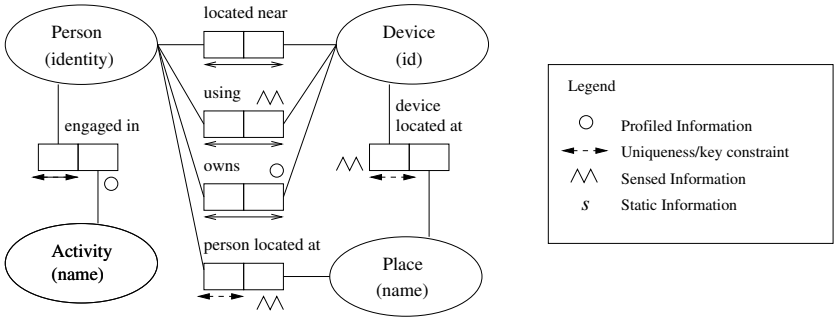


Fig. 1. A sample context model

have to be specified once. This can be achieved by separating the granularity preferences from the privacy preferences.

In the remainder of the paper we present an obfuscation method that can meet these challenges. We use ontologies to describe how different instances of an object type relate to one another with respect to granularity. This information is used to find lower granularity instances of an object type during obfuscation. Context owners can control the obfuscation procedure using our preference language for specifying constraints on granularity. In the event that obfuscation is not possible, the release of a value of "unknown" is used to provide the system with plausible deniability.

3 The sample context model

In this section we present a sample context model that is used later to demonstrate the functionality of our obfuscation mechanism and preference model extensions. While we use the model extensively for demonstrating our approach, our approach is not tied to the model, and is general enough to be used to provide obfuscation in other context management systems.

The sample model was constructed using the Context Modelling Language (CML). CML was developed as an extension of the Object Role Modelling method (ORM) [6], with enhancements to meet the requirements of pervasive computing environments. A more rigorous discussion of CML is provided in [7].

The sample context model is graphically represented in Figure 1. It models four object types: Activities, Devices, Persons, and Places. These are depicted as ellipses, with the relationships between them shown as boxes. Each box is labelled with the name of the relationship, and annotated with any contraints on that relationship. With reference to terminology, instances of object types are referred to as objects. Relationships between object types are captured as fact types, and relationships between objects are captured as facts.

According to our example model, a Person can be modelled as being at a particular Place through the locatedAt fact type. Similarly, a Person may interact with a Device which they may own, use and be located near. The respective fact types for these are own, using and locatedNear. Two other fact types, engagedIn

and locatedAt, model the current Activity of a Person and a Person's location, respectively.

Using a previously developed technique, the context model is mapped onto a context knowledge base [8] such that fact types are represented as relations. Context information in the database is stored as context facts or tuples, which express assertions about one or more objects. The context information can be accessed by directly querying facts, or through the use of situations. These situations are a high-level abstraction supported by the modelling technique we use from [7]. Situations are defined using a variant of predicate logic and can easily be combined using the logical connectives *and*, *or* and *not*, as well as special forms of the existential and universal quantifiers. The example situation below, EngagedInTheSameActivity, takes two people as parameters and returns true if they are working on the same activity.

EngagedInTheSameActivity(person1, person2):
$\exists$*activity*,
•engagedIn[person1, activity],
•engagedIn[person2, activity].

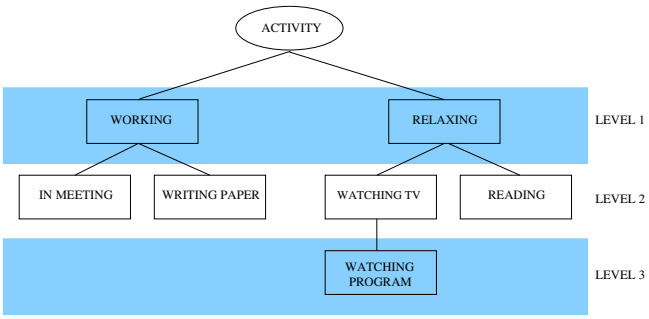
Our context model captures ownership of context information to enable the CMS to determine whose preferences should be applied. The ownership definitions we use for this also enable the resolution of overlapping ownership claims that arise in pervasive computing. These can occur when multiple parties claim jurisdiction over context information. Our scheme, presented in [9], models ownership at both the fact and situation levels, such that facts and situations can have zero, one or multiple owners. Facts and situations with no owners are considered public, and are always visible. Facts and situations that have owners are considered non-public, are always visible to their owners, but are only disclosed in accordance with the owner's privacy preferences to third parties. The specification of ownership directly on situations is necessary for efficiency, as situations can potentially operate on context owned by multiple parties. Performing ownership tests on each context fact is extremely inefficient, particularly when the situation involves large numbers of context facts, all with different owners.

The following section discusses our novel use of ontologies to provide obfuscation of context information.

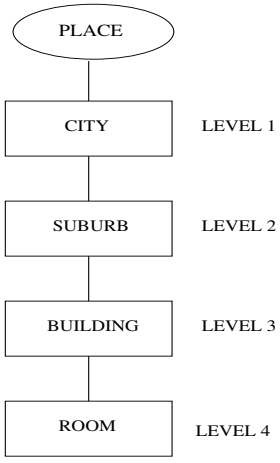
4 Ontologies for obfuscation

In our new approach to obfuscation, each object type in the context model is described with an ontology. These ontologies are constructed from what we refer to as "ontological values", which are arranged in a hierarchy. The hierarchy represents the relative granularity between the values, with parent ontological values being more general than their child values. To determine an object's granularity, the obfuscation procedure matches it with an ontological value. Objects that match to the ancestors of this ontological value are of lower granularity than the

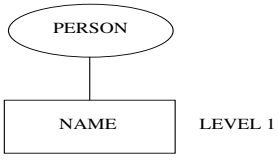
object. Conversely, objects which match to the children of the ontological value are considered to be of higher granularity. Ontological values at the same level in the hierarchy are considered to be of the same granularity. Example ontologies for Activity, Person and Place are provided in Figure 2.



(a) Activity Ontology



(b) Place Ontology



(c) Person Ontology

Fig. 2. Ontologies for obfuscation

When context facts are obfuscated, the CMS decomposes them into their component objects. These objects are then matched with ontological values from their respective object type's ontology. The behaviour of the system then branches, depending on the nature of the ontology being used. We have identified two different classes of ontology, which we refer to as Class I and Class II.

The first class of ontology is constructed from ontology values which are themselves valid instances of the object type. As such, the ontology values can be used as the result of obfuscation. These ontologies we refer to as Class I ontologies. In the second class of ontology, referred to as Class II ontologies, this is not the case. The ontology value must be matched with an actual instance from the context knowledge base.

The ontology for Activity is an example of a Class I ontology. Thus, the ancestor value of "reading", i.e "relaxing", is a valid Activity object. A potential obfuscation of engagedIn[Alice, reading] is then engagedIn[Alice, relaxing].

Place and Person are examples of object types with Class II ontologies. With respect to Person, the "name" ontological value is not a valid Person object, and must be matched with an entry in the context knowledge base before being used. Likewise, the ontological values in the Place ontology, e.g. "city" and "suburb", need to be matched to objects in the context knowledge base. Obfuscating a person's location to locatedAt[Bob, city], is meaningless. A meaningful result would be locatedAt[Bob, Brisbane], where an instance of Place has been matched to the "city" ontology value.

The approach presented in this paper considers both the Class I and Class II ontologies to be system-wide and largely static. We are currently working on extending the system to enable per user customisation of the ontologies.

5 Specification of preferences

In our approach the privacy policy of a context owner is captured as a set of preferences. The preference language we have developed to facilitate this supports two kinds of preferences for controlling the release of context information: the *privacy preference*, which limits the disclosure of context information to third parties; and the *granularity preference* which the context owner can use to control the level of detail of disclosed context information.

5.1 Privacy preferences

Privacy preferences enable context owners to authorise or deny disclosure of context information based on a set of activation conditions. To express these preferences we use an extended version of a preference model we developed previously for context-aware applications [7].

In our privacy preference model, each preference is given a unique name. The activation conditions for the preference are then listed as part of a scope statement, which is declared beginning with a "when". These activation conditions can be specified over fact types, situations and conditions on parameters like *requester* or *purpose*. If all the access conditions are met, the preference activates and either ratifies or prohibits the disclosure. These two options are represented as ratings of oblige or prohibit, respectively. An example privacy preference for an entity called Alice is given below:

```
privacy_pref = when locatedAt[Alice, Home] AND
                    requester = Bob AND
                    purpose = Advertising AND
                    factType = engagedIn
               rate prohibit.
```

This preference activates when Alice is located at home, and Bob requests the activity that Alice is currently engaged in, which he intends to use for advertising purposes. As this is an example of a negative preference, when all the activation conditions are met, the preference prohibits the disclosure of the queried information. Not all of the four conditions in the example preference need to be present. By removing conditions, the preference is made more general, e.g, removing "purpose" and "factType" will cause the preference to block all requests from Bob when Alice is at home.

The example preference can be converted into a positive preference by replacing the prohibit rating with oblige. This flexibility to specify both positive and negative preferences is lacking in other preference languages such as that used by Gandon and Sadeh [5]. Both positive and negative are desirable, as they make preference specification considerably easier for users [1].

5.2 Granularity preferences

In our novel approach to obfuscation, context owner control over the obfuscation procedure is provided with the granularity preference. This new preference is separated from the privacy preference as it operates on instances of objects rather than on facts. As a result of this, one granularity preference can cover all assertions in which instances of the protected object type participate. This is an improvement over other obfuscation techniques, like [5], where an obfuscation rule only operates on one assertion.

With regard to structure, the granularity preference definition begins with the preference name, and then lists the activation conditions in the scope statement. As the format for the granularity and privacy preference scope statement is the same, it will not be repeated here.

An example granularity preference is given below:

```
granularity_pref = when requester = Bob
                   on Activity
                   limit level 1
```

When the activation conditions specified in the scope are met, the specified granularity limit becomes active on the object type. While the granularity preference is active, context facts containing instances of the protected object type are only disclosed if the protected instances are less than or equal to the granularity limit. Instances with a granularity higher than the limit are obfuscated to meet the granularity limit.

The limit can be specified in two ways: as a cut-off; or as a granularity level number. The cut-off method uses a value in the ontology below which disclosure

is not permitted. Consider the Activity ontology. If a cut-off of "watching TV" was specified on Activity, any objects matching "watching program" could not be disclosed in response to queries, while both "watching TV" and "relaxing" could be disclosed freely.

The second means states the limit as a granularity level number taken from the relevant ontology. The level numbers start from 1 and increase up to the highest level of granularity supported in the ontology for that object type.

For a comparison between the two approaches, consider Figure 2. A granularity level of 1 would allow the release of objects which match with the "working" and the "relaxing" ontological values. Access to other Activity objects would be blocked as all other Activity objects have a granularity in excess of level 1. In contrast, a cut-off of "watching TV" prevents access to "watching program" only. If the "watching TV" cut-off granularity preference were active, and the user were engaged in a work-related activity, (i.e, not relaxing), the granularity preference would have no effect.

6 Preference evaluation

A context information query on the context knowledge base issued by a third party causes the CMS to evaluate the current context. If the requested context information is available, the CMS determines whether or not the context is owned by checking the context ownership definitions, which are described in more detail in an earlier paper [9]. If the information is public, meaning there are no owners, the request is granted. If the request was submitted by an entity with ownership rights to the context, the request is permitted. If the requester is not an owner, the owner's preferences are retrieved and evaluated by the CMS.

6.1 Evaluation of privacy preferences

The privacy preferences of the owners are evaluated using the current context information in the CMS. If any of the preferences specified by the owner prohibit the disclosure, the evaluation is aborted, and an "unknown" value is returned. If the collective privacy preferences of all the owners permit the disclosure, any granularity preferences specified by the context owners are retrieved and evaluated.

6.2 Evaluation of granularity preferences

Provided that disclosure is authorised, the granularity preferences of the owners are then evaluated. If any of the granularity preferences are activated by the current context, the release of all context facts containing an instance of the object type on which the granularity preference operates become subject to obfuscation. The most restrictive granularity preferences are used to obfuscate the context information.

To obfuscate a context fact, it is necessary to obfuscate its component objects. To achieve this, the objects must be extracted from the fact and matched

to ontological values in their object type's ontology. Obfuscation then involves selecting an ancestor of this value from the hierarchy that satisfies any disclosure limitations specified by the context owner(s). The process then diverges, depending whether the ontology is Class I or Class II.

6.3 The affect of obfuscation on situations

When evaluating situations it may be necessary to access context facts which refer to specific objects. If the object is protected from disclosure by granularity preferences, then the situation cannot be evaluated. An example of this can be seen by considering the situation InBrooklynWithAlice below. This situation takes a person object as its parameter and returns true if the person is in the suburb of Brooklyn at the same time as Alice.

InBrooklynWithAlice(person):
locatedAt[Alice, Brooklyn] and
locatedAt[person, Brooklyn].

This situation requires specific locatedAt context facts from both Alice and the parameter *person*. If either of these people have granularity preferences which prevent release of Place instances at the suburb level (i.e., Brooklyn), the situation cannot be evaluated, and will return *unknown*.

6.4 Failure of the obfuscation process

The two classes of ontology defined for obfuscation behave quite differently during the obfuscation process. With Class I ontologies the ontology values are themselves valid instances of the object type, and so obfuscation will never fail to find a return value. However, with Class II ontologies, it is possible to have ontological values for which there is no match in the context knowledge base. To overcome this, the obfuscation procedure attempts to obtain a more general value from the hierarchy by recursively testing to see if any of the ancestor values are present. Obfuscation fails if the root of the hierarchy is reached and a match has not been made. Should this occur, the *unknown* value is returned.

7 Example scenario

In this section we present an example to demonstrate the operation of the obfuscation mechanism. The granularity preferences in the example make use of the Activity ontology depicted in Figure 2.

A person, Alice, defines a set of preferences for the disclosure of her context information. The preferences relevant to this example are given below.

```
p1  = when requester = Bob
      rate oblige
```

```
g1 = when requester = Bob
     on Activity, Place
     limit level 1
```

Through privacy preference p1, Alice allows Bob access to her context. However, she places constraints on the level of detail available. If Bob issued a query on Alice's current activity, the query would be processed by the CMS which would consult the ownership definitions to determine that Alice is the owner of the context information. The CMS would then apply Alice's privacy preferences, which in this case permit the release of information to Bob. Any granularity preferences specified by Alice would then be evaluated by the CMS. In this case, preference g1 limits Bob's access to instances of Activity and Place so that he may only access them up to granularity level 1. If Alice is currently engagedIn[Alice, Watching_Program], the CMS would extract the object instances and obfuscate those of type Activity. In this instance Watching_Program is of granularity level 4. By examining the ontology for Activity, the value would be obfuscated to granularity level 1, Relaxing. The result returned to Bob would then be engagedIn[Alice, Relaxing].

Not to be thwarted, Bob knows that Alice always carries around her PDA device. From the location of her device, he could possibly infer more detail about her activity. To obtain this information he queries the location of Alice's PDA. Again, privacy preference p1 permits the access. However, granularity preference g1 activates and obfuscates the location of the PDA to granularity level 1. All that Bob learns is the city in which Alice's PDA is located, i.e., locatedAt[Alice_PDA, Brisbane]. This information is of no help to Bob.

The power of the approach presented in this paper lies in the separation between privacy preferences and granularity preferences which operate on object types. The context owner defines their granularity requirements for object types. These granularity requirements are then applied to all instances of the object types over which the owner has jurisdiction. Ownership rights, or jurisdiction, are then determined by the CMS based on the ownership definitions provided in the context model. In our example, Alice did not have to specifically specify limitations on Bob accessing her device location, as all location information belonging to Alice was protected by preference g2.

8 Conclusions and future work

In this paper we presented an obfuscation mechanism to enhance user privacy. The mechanism is capable of modifying the granularity of context information to meet limits specified by the owners of the context information regarding its disclosure. Unlike other obfuscation mechanisms, ours supports variable levels of obfuscation for arbitrary types of context information. The obfuscation procedure is supported by ontological descriptions of object types. These descriptions capture the relative granularities of object type instances in a hierarchical fashion. Users control the obfuscation procedure by specifying granularity preferences which control the extent to which context information is obfuscated before being released to those who request it. Unlike the solutions presented in [5] and [2], the granularity restriction is not tied to a particular privacy preference, but rather is specified on an object type from the context model. This means that a general granularity preference can be reused across many different privacy preferences.

We are currently exploring ways to provide per user customisation of ontologies. This will enable users to more closely tailor the obfuscation mechanism to their needs. We are also developing a preference specification tool to conduct usability testing on our approach.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *Proceedings of the twelfth international conference on World Wide Web*, pages 629–639. ACM Press, 2003.
2. H. Chen, T. Finin, and A. Joshi. A Pervasive Computing Ontology for User Privacy Protection in the Context Broker Architecture. In *Technical Report TR-CS-04-08*, Baltimore County, Maryland, USA, 2004. University of Maryland.
3. L. Cranor, M. Langheinrich, and M. Marchiori. *A P3P preference exchange language 1.0 (APPEL1.0)*, April 2002. http://www.w3.org/TR/P3P-preferences; last accessed 4/11/2004.
4. L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, 2001. http://www.w3.org/TR/P3P/; last accessed 4/11/2004.
5. F.L. Gandon and N.M. Sadeh. A Semantic e-Wallet to Reconcile Privacy and Context Awareness. In *The 2nd International Semantic Web Conference (ISWC2003)*, pages 385–401, Sanivel Island, Florida, USA, 2003. Springer-Verlag Heidelberg.
6. T.A. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufman, San Francisco, 2001.
7. K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In *2nd IEEE Conference on Pervasive Computing and Communications, PerCom'04*, Orlando, March 2004.
8. K. Henricksen, J. Indulska, and A. Rakatonirainy. Generating Context Management Infrastructure from Context Models. In *Mobile Data Management 2003 (MDM2003)*, 2003.
9. K. Henricksen, R. Wishart, T. McFadden, and J. Indulska. Extending context models for privacy in pervasive computing environments. In *Proceedings of CoMoRea'05 (to appear)*, 2005.
10. J.I. Hong and J.A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189. ACM Press, 2004.
11. C. Johnson, D. Carmichael, J. Kay, B. Kummerfeld, and R. Hexel. Context Evidence and Location Authority: the disciplined management of sensor data into context models. In *Proceedings of the first International Workshop on Context Modelling, Reasoning and Management at UbiComp2004*, pages 74–79, September 2004.
12. S. Lederer, C. Beckmann, A. Dey, and J. Mankoff. Managing Personal Information Disclosure in Ubiquitous Computing Environments. Technical Report IRB-TR-03-015, Intel Research Berkley, 2003.