

# Context Privacy and Obfuscation Supported by Dynamic Context Source Discovery and Processing in a Context Management System

Ryan Wishart<sup>1,2</sup>, Karen Henricksen<sup>2</sup>, and Jadwiga Indulska<sup>1,2</sup>

<sup>1</sup> School of Information Technology and Electrical Engineering,  
The University of Queensland

and

<sup>2</sup> NICTA \*

{wishart,jaga}@itee.uq.edu.au, Karen.Henricksen@nicta.com.au

**Abstract.** The extensive context information collection abilities of ubiquitous computing environments represent a significant threat to user privacy. In this paper we address this threat by introducing a context information privacy mechanism. Our approach relies on context-dependent ownership definitions and context owner-specified privacy preferences to control context disclosure to third-parties. These privacy preferences enable context owners to stipulate not only to whom their context information can be disclosed and the conditions of disclosure, but also the level of detail at which the context information can be disclosed. Context information that cannot be disclosed at its existing level of detail is obfuscated to meet detail level requirements stipulated by its owner. To achieve this obfuscation of context information we introduce a new approach based on dynamic discovery and processing of context sources. Our new approach is demonstrated in a Context Management System in which context source discovery and processing is facilitated by the SensorML sensor description standard being developed by the Open Geospatial Consortium.

## 1 Introduction

Potential users of ubiquitous computing environments frequently cite privacy as a major concern [1] in their adoption of the technology. These user privacy concerns arise from both the degree and the sensitivity of the context information collected by the ubiquitous computing environment.

To address these concerns it is necessary to provide a context information privacy mechanism that can (1) determine the owner of particular context information, and (2) dynamically handle the disclosure of the context information according to that owner's preferences.

---

\* NICTA is funded by the Australian Government's Department of Communications, Information Technology, and the Arts; the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs; and the Queensland Government.

Here we consider the owner of context information to be the person, or organisation, that determines how that context information should be used and when it can be disclosed to other parties.

With regard to the first of the requirements, existing work in the field either does not explicitly address the issue of ownership or, if it does, ownership is statically allocated to a particular entity (often the producer of the context information). This does not correlate with ownership in ubiquitous computing environments which may be context-dependent and involve multiple parties [2].

In addressing the second requirement, context-dependent privacy preferences can be used to capture the context information owner’s disclosure preferences. While several systems supporting such preferences exist (such as PawS [3] and the Semantic e-Wallet [4]) they provide only limited control over the level of detail/granularity at which context is disclosed. To meet disclosure detail level requirements for context information, obfuscation mechanisms are required. In existing systems these obfuscation mechanisms are restricted to obfuscating particular types of context information (e.g., location), returning preset values as the result of obfuscation, or supporting a set number of levels of detail for all types of context information.

To address the deficiencies in existing work, we present a context information privacy mechanism that (1) supports context-dependent, multi-party ownership of context information, (2) enables context information owners to control to whom and the level of detail with which their context information is disclosed and (3) provides a general obfuscation mechanism that can dynamically adjust the detail level of context information to meet the disclosure requirements of the context information’s owners.

A Context Management System (CMS) is used to demonstrate our general obfuscation mechanism. This CMS is able to automatically discover new context sources, or perform processing of stored sensor output, to fulfil the detail level required by any preferences the context information owner may have specified. The CMS’s ability to perform this automatic discovery leverages off the Open Geospatial Consortium’s SensorML sensor description specification [5].

The remainder of the paper is structured as follows. We first motivate our research with an overview of related work in the field (given in Section 2). In Section 3 we present background information on the SensorML specification, as well as our context modelling approach and privacy preference mechanism. Obfuscation of context information is discussed in Section 4 before we then present an architecture for a CMS capable of dynamic context source discovery and processing in Section 5. An explanation of how this CMS can be used to support our context information privacy and obfuscation approach is given in Section 6. Concluding remarks for the paper are then made in Section 7.

## 2 Related Work

In this related work section we provide an overview of the context information privacy literature focusing on support for context information obfuscation.

The Confab toolkit, developed by Hong and Landay [1], enables the expression of complex, context-dependent privacy policies for context information in a ubiquitous computing environment. In addition, a built-in obfuscation mechanism allows the release of this context information at different levels of detail. A significant shortcoming of the approach is that the system operates on the tacit assumption that producers of context information are its owners. This is not always the case with ubiquitous computing environments where ownership may be multi-party and context-dependent [6].

Lederer et al. [7] present an approach to context information privacy which allows users to define sets of disclosure preferences in terms of a few predefined types of context information. The system provides limited obfuscation of context information to a maximum of four levels of detail (precise, approximate, vague and undisclosed). However, these four levels are not appropriate for all types of context information; some types of context cannot be obfuscated at all (as in the case of simple boolean or on/off values), while other types can be meaningfully disclosed at more than four levels of detail.

A different approach to context information privacy in ubiquitous computing environments was developed by Gandon and Sadeh [4] for their Semantic eWallet privacy management system. The approach supports detailed, context-dependent preference specification. Obfuscation of context information is also provided. However, the obfuscation is extremely static in the sense that users predefine the values disclosed by the system in a case-by-case fashion. Unfortunately, if the environment changes, the statically-defined values may become meaningless or unusable.

Rather than actively protect context information using a rule-based approach, the pawS system [3] seeks to make the user aware of the context gathering infrastructure available within a particular location. The system operates by notifying users using a privacy beacon wherever context information is collected. This privacy beacon transmits data usage policies that explain which context information will be collected and how the collector intends to use that information. The pawS system is capable of providing information at different levels of detail but cannot obfuscate a particular item of context to a lower level of detail. The concept of ownership is also not directly addressed as there is no mechanism for specifying which context information belongs to a user.

From this review of the related work it is clear that existing privacy mechanisms (1) do not support context-dependent, multi-party ownership as is found in real ubiquitous computing scenarios, and (2) lack or provide only limited support for obfuscation of context information. Those that do support obfuscation of context information support obfuscation on a limited number of context types or restrict the number of levels of obfuscation that can be applied.

## 3 Background Information

### 3.1 SensorML for Automated Sensor Discovery and Configuration

In recognition of the growing heterogeneity of sensor networks, the Open Geospatial Consortium (OGC) is developing a common format for describing sensor functionality, their outputs (referred to as observations) and how to process these observations. It is intended that such descriptions would be made available online along with sensor observations, allowing applications to automatically discover and use remote sensors.

The XML syntax and semantics for sensor descriptions are covered in the SensorML specification [5], while mapping sensor output to SensorML Observations is discussed in the Observations and Measurements proposal of the OGC[8]. SensorML also supports Process Chains, a mechanism for describing the inputs, outputs and parameters of executable programs that can be used to transform sensor data into more meaningful results. As with the SensorML descriptions of sensors and sensor data that can be dynamically discovered online, Process Chains can also be found and loaded at runtime, enabling dynamic processing of sensor data.

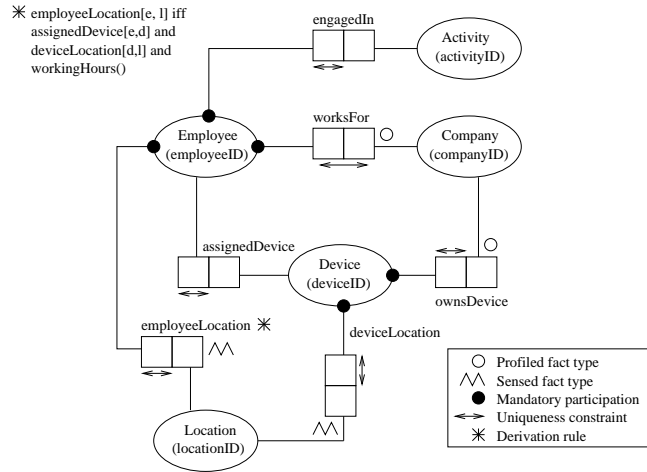
As discussed in [9], the wide-spread adoption of emerging standards like SensorML would enable a ubiquitous computing environment to automatically locate new sensors, interpret their output (observations) and, by applying SensorML Process Chains, automatically perform any processing of the sensor observations necessary to make them compatible with the context information required by applications in the ubiquitous computing environment.

### 3.2 Context Modelling and Ownership

Our context information privacy mechanism requires a context model of the ubiquitous computing environment to operate. In the Context Modelling Language (CML) [10] we use to construct our context model, major entities in the environment are captured as object types (e.g., Employee, Location, Activity). Instances of the object types, such as the employee Alice, or the location Brisbane, are referred to as objects. Relationships between object types are modeled as fact types, while relationships between objects are represented as facts.

An example context model, depicted using the graphical notation for CML, is given in Figure 1. This model has several object types: Employee, Company, Location, Device (representing wireless computing devices) and Activity.

In the example context model, companies assign wireless computing devices to their employees. Both the wireless computing devices and the employees have a location, while the employees also have an activity (such as “meeting clients”, or “using the computer”). In the context model, the location of employees can be derived from the location of devices according to the derivation rule specified in Figure 1. Within this rule “workingHours()” is a situation that returns true if the current time is within working hours and false otherwise. More information on situations can be found in [10].



**Fig. 1.** An example context model for a company that tracks the location of its employees using wireless computing devices that it assigns to each employee

The context model is mapped onto database schema such that fact types are represented as relations. Context information in the database is stored as context facts, which express assertions about one or more objects. An example context fact is **worksFor[Alice, MiracleMart]**. The fact type in this example is “worksFor”, while Alice is an instance of the Employee object type, and MiracleMart an instance of the Company object type.

To make the context model ownership-aware, we introduced an ownership modelling approach in [2]. Our approach recognises that some of the object types in the context model (referred to as *first class* object types) have the capacity to act as owners of the context information related to them. Examples of this type include Employee and Company object types.

Other objects types, such as computing devices and physical places, fall under the ownership of various first class object types. These are *second class* object types. The associations between first and second class objects types can be context-dependent; therefore, we state their ownership using one or more fact types. A detailed discussion of the vocabulary used for declaring this ownership is outside the scope of this paper, but more information can be found in [2].

The ownership model also supports *third class* object types, which are any object types that require no owners.

Our approach also supports ownership specification at the level of the fact type. This enables each fact type to have its own rules for assigning ownership. These rules associate each fact (i.e. an instance of a fact type) with zero, one or multiple owners. Facts that have zero owners are *public* and can be freely disclosed to anyone.

### 3.3 Privacy Preference Language

In our approach, context information owners are able to express their privacy requirements using context-dependent privacy preferences. These preferences can be defined for certain contexts, and then combined to form a comprehensive privacy policy. Due to space restrictions, we provide only a brief overview of the privacy preference language.

Our privacy preference language supports two preference types: binary privacy preferences that allow disclosure to be either granted or denied, and granularity preferences that enable context owners to specify detail level restrictions on the disclosure of their context facts.

**Binary Privacy Preferences** Binary privacy preferences are structured as a unique identifier, a scope statement and a rating. The scope defines the context in which the preference applies, in terms of fact types and conditions on parameters of the current access request (most importantly, the identity of the *requester*). The rating indicates whether the preference is to allow or deny the disclosure of the context information.

```
alice_preference_1 = when NOT worksFor[requester, MiracleMart]
                    rate deny
```

An example binary preference is given above that denies access to all Alice's context information if the requester does not work for MiracleMart.

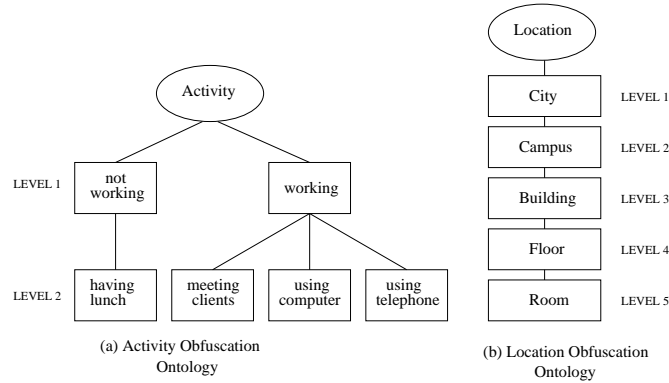
**Granularity Preferences** Granularity preferences enable context information owners to control the obfuscation of their context information. These preferences are linked to object types in the context model, and apply to context facts containing instances of the linked object type.

To support the obfuscation process, we rely on predefined ontologies for each object type, which we refer to as obfuscation ontologies. These consist of *ontology values* arranged by relative detail into a hierarchy such that parent nodes in the hierarchy are more general than their children. Simplified obfuscation ontologies for the Activity and Location object types are shown in Figure 2.

Granularity preferences contain a scope statement, the object type to which the granularity preference is linked, and a detail limit that context facts containing the instances of the linked object type must meet before those context facts can be disclosed. This limit is expressed as a level number in the obfuscation ontology of the linked object type (e.g., detail level 1). An example granularity preference is given below for the employee Alice:

```
alice_preference_2 = when equals(requester, Bob) AND workingHours()
                    on Activity
                    limit level 1
```

This preference stipulates that any context information disclosed to the requester Bob during working hours that contains instances of Activity should be at most detail level 1. In this example **equals** and **workingHours** are *situations* as discussed briefly in Section 3.2.



**Fig. 2.** Simplified obfuscation ontologies for the Activity and Location object types

```

City:
  Brisbane
  Campus:
    Brisbane Campus
    Building:
      MiracleMart Main Office
      Floor:
        floor 6
        Room:
          78-633
          78-625
          78-621
        floor 4
        ...

```

**Fig. 3.** An excerpt from a taxonomy for the Location object type that shows how valid Location objects are related in terms of detail level

## 4 A General Obfuscation Mechanism for Context Information

Our obfuscation mechanism is activated when a context owner’s granularity preferences require a context fact to be reduced in detail. This detail reduction process is performed on objects within the context fact using obfuscation ontologies. Two kinds of obfuscation ontologies are recognised by the mechanism: Class 1, and Class 2.

We address obfuscation of objects whose object types have a Class 1 obfuscation ontology first, using the context fact **engagedIn[Alice, using computer]** as an example. For this context fact, we assume “using computer” must be obfuscated to a lower detail level. As we are dealing with a Class 1 obfuscation ontology, the ontology values in the obfuscation ontology are themselves valid object instances [6].

To generate a new context fact the obfuscation mechanism needs to (1) locate the current ontology value (“using computer”) in the obfuscation ontology and

then (2) traverse up the hierarchy until an ontology value of the required detail level is reached. For example, if the required detail level is level 1, and “using computer” has detail level 2 (based on Figure 2), the obfuscation mechanism would traverse up the hierarchy of ontology values to reach the detail level 1 ontology value “working”.

The obfuscation mechanism is then able to generate the new context fact **engagedIn[Alice, working]**.

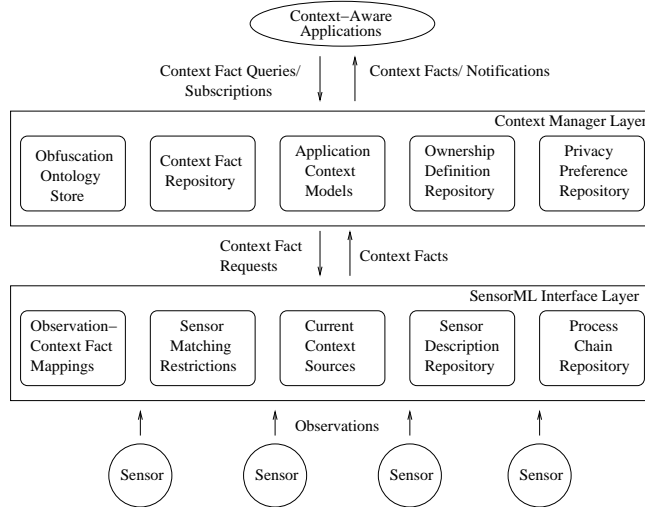
To demonstrate the obfuscation of objects with a Class 2 obfuscation ontology we use the example context fact **employeeLocation[Alice, 78-633]**. In this example the “78-633” object is an instance of the Location object type, and is of detail level 5 (“78-633” corresponds to the ontology value “Room” in the taxonomy shown in Figure 3. “Room” is of level 5 detail according to the obfuscation ontology for Location in Figure 2).

To obfuscate objects using a Class 2 obfuscation ontology requires that the obfuscation mechanism understand how each object instance is related (i.e. the system must know that 78-633 is on floor six within the MiracleMart Main Office in Brisbane). This can be achieved using a top-down approach in which a knowledgeable system designer specifies detailed taxonomies describing how particular object instances are related (such as the taxonomy given in Figure 3).

An alternative approach to using a large, detailed taxonomy of object instances is to split up the knowledge in the taxonomy into a large number of rules (e.g., one of these rules could be ‘all rooms with room numbers matching “78-6..” are located on the 6th floor’).

The taxonomy approach to obfuscation of context facts, which we discussed first, suffers from scalability issues as the relationship between object instances for each object type must be pre-specified. As the ubiquitous computing environment is highly dynamic, generating an exhaustive taxonomy for all possible object instances of each object type in the context model is infeasible. This means that non-exhaustive taxonomies must be used. However, if the ubiquitous computing environment changes, the non-exhaustive taxonomy must be re-specified.

In comparison, the bottom-up rule-based approach does not require large, detailed taxonomies to be specified by a knowledgeable designer. Rather, the system dynamically discovers rules (potentially implemented as declarative rules or as data processing programs) to process the context facts to the desired level of detail. This gives it the ability to adjust to significant changes in the ubiquitous computing environment (such as when a user moves location) that a non-exhaustive taxonomy of object instances cannot handle. In the following section we describe a Context Management System that performs obfuscation using this bottom-up rule-based approach (implemented using executable programs accompanied by SensorML Process Chain descriptions).



**Fig. 4.** An example layered architecture for a Context Management System implementing our context information privacy mechanism

## 5 Architecture of the Context Management System

To demonstrate our privacy approach, including our obfuscation mechanism, we introduce a layered Context Management System (CMS) based on that of Indulska et al. [9]. Figure 4 shows the layered structure of the CMS.

The architecture is composed of three layers: a Sensor Layer, a SensorML Interface Layer and a Context Manager Layer.

The Sensor Layer represents the software and hardware sensors within the ubiquitous computing environment that emit SensorML Observations. These observations are fed into the SensorML Interface Layer.

The SensorML Interface Layer (SIL) performs two important tasks. Firstly, it processes observations from the Sensor Layer before converting them into context facts using the Observation-Context Fact Mappings. Secondly, it locates new sources of context information by searching the Sensor Description Repository for sensors using the Sensor Matching Restrictions. The Sensor Description Repository is where SensorML descriptions of active sensors are stored. The Sensor Matching Restrictions are selection rules for choosing new sensors (such as sensor type or accuracy) as well as restrictions on where the sensor is (e.g., geographical location). A Process Chain Repository is also maintained that stores currently used Process Chains and their associated executable programs.

In the Context Manager Layer context facts provided by the SensorML Interface Layer are stored in a Context Fact Repository. Application Context Models for context-aware applications in the ubiquitous computing environment are also stored. These are used to interpret the stored context facts on a per application basis. The Obfuscation Ontology Store, the Ownership Definition Repository

and the Privacy Preference Repository are also located in the Context Manager Layer. The Obfuscation Ontology Store houses the obfuscation ontologies used by the obfuscation mechanism. The Ownership Definition Repository and the Privacy Preference Repository hold the ownership definitions and the privacy preferences for our context information privacy mechanism, respectively.

## 6 The Privacy and Obfuscation Mechanism

The privacy functionality of the Context Management System operates as follows. Context-aware applications issue context fact queries (or requests for notification of context changes) to the Context Manager Layer. Each query (or request) is first evaluated to determine (1) the owner of the requested context fact, (2) whether or not the requested context fact is present in the Context Fact Repository, and (3) if the owner has preferences pertaining to the disclosure of the requested context fact.

If the requested context fact is public (i.e. it has no owner), then the request is permitted. Should the requested context fact be owned (as per the ownership definitions), then the identity of the requester is checked to see if it is one of the owners. If it is an owner, then the request is permitted as our ownership mechanism guarantees access to all owners of a context fact. If the requester is not an owner, then the preferences of the owners are evaluated to determine whether or not the context fact should be disclosed. Provided the disclosure is permitted, any relevant granularity preferences are then evaluated. If the granularity preferences require obfuscation of the context fact prior to disclosure this obfuscation process occurs.

Obfuscation of context facts associated with Class 1 obfuscation ontologies occurs as described in Section 4.

When the obfuscation of an object associated with a Class 2 obfuscation ontology is required (such as when the context fact **employeeLocation[Alice, 78-633]** must be obfuscated to detail level 1) the following occurs.

The Context Manager Layer first checks the context model for any additional information about the context fact. In our example, **employeeLocation[Alice, 78-633]** is marked as derived from the location of **WirelessDevice1** (the device assigned to Alice). To obfuscate the location of Alice requires the location of the device be obfuscated. To achieve this, the Context Manager Layer issues an obfuscation request to the SIL for **deviceLocation[WirelessDevice1, 78-633]**, with the Location object obfuscated to detail level 1 (i.e. city level).

The SIL notes that the current Observation to context fact mappings produce deviceLocation context facts of detail level 5. The SIL re-examines its mappings to determine if an alternative context source is available that provides SensorML Observations which can be mapped to a context fact with the required detail level. If the SIL fails to find an alternative context source, it locates the SensorML Observation that produced the context fact **deviceLocation[WirelessDevice1, 78-633]**. This SensorML Observation, a location report for “WirelessDevice1”, is shown in Figure 5.

```

<om:Observation gml:id="Observation1">
...
  <om:featureOfInterest
    xlink:href="http://MiracleMart.org/Devices/instances/WirelessDevice1"/>
  <om:observedProperty
    xlink:href="http://MiracleMart.org/DeviceLocation"/>
  <om:result xsi:type="swe:ScopedNameType"
    codeSpace="http://MiracleMart.org/Offices/">
    78-633
  </om:result>
...
</om:Observation>

```

**Fig. 5.** Excerpt from a SensorML Observation giving the location of WirelessDevice1 as office *78-633*

The SIL then searches through the Process Chain descriptions in the Process Chain Repository looking for a program capable of converting the SensorML Observation in Figure 5 to a SensorML Observation of the required city-level detail. It is assumed that each Process Chain description includes information (such as the format and detail level) on the inputs and outputs of the associated program.

If no single program is able to do the conversion, the SIL attempts to chain multiple programs together to obtain a SensorML Observation with the required detail level. If this also fails, the SIL then conducts an online search for suitable programs. If this search also fails, the SIL cannot proceed further and the obfuscation process is aborted. This means that the context information cannot be disclosed.

For this example we assume that the SIL is successful in its search, and uses the program(s) it found to convert the SensorML Observation in Figure 5 to the required detail level.

This new SensorML Observation is then mapped using the Observation to Context Fact Mappings maintained by the SIL to produce a new context fact **deviceLocation[WirelessDevice1, Brisbane]**. This is returned to the Context Manager Layer, where the derivation rule “During working hours, Alice’s location is that of the device that she is assigned” (defined in Figure 1) is applied to obtain the context fact **employeeLocation[Alice, Brisbane]**. The Context Manager Layer then releases the context fact to the context-aware application that queried (or requested) it.

## 7 Conclusion

In this paper we presented a context information privacy mechanism designed to protect the privacy of context information owners in the ubiquitous computing environment. Unlike existing approaches, our solution provides explicit support for context-dependent, multi-party ownership. In our approach owners of context information are able to specify the conditions under which their context information is disclosed, including the detail level at which disclosure may take

place. We presented a context obfuscation mechanism able to dynamically adjust the level of detail of context information to meet any detail level disclosure requirements specified by the context owner(s). Unlike existing approaches to obfuscation of context information, our approach is not restricted to particular context types (such as location), does not return pre-specified values as the result of obfuscation, and does not restrict obfuscation to a static number of detail levels for all context types. Two schemes for implementing our obfuscation mechanism were discussed: using taxonomies, and using a bottom-up rule-based approach in which the data processing rules used for obfuscation of context facts can be implemented as dynamically discoverable programs. We then described a Context Management System incorporating our context information privacy mechanism's functionality. This Context Management System is able to dynamically discover new sources of context information, and new executable programs (which are accompanied by SensorML Process Chain descriptions) to process the information from context sources, enabling it to obfuscate context information without the need for detailed taxonomies.

## References

1. Hong, J., Landay, J.: An architecture for privacy-sensitive ubiquitous computing. In: 2nd International Conference on Mobile Systems, Applications, and Services, ACM Press (2004) 177–189
2. Henricksen, K., Wishart, R., McFadden, T., Indulska, J.: Extending context models for privacy in pervasive computing environments. In: 2nd International Workshop on Context Modelling and Reasoning (CoMoRea), PerCom'05 Workshop proceedings, IEEE Computer Society (2005)
3. Langheinrich, M.: A privacy awareness system for ubiquitous computing. In: Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2002). (2002)
4. Gandon, F., Sadeh, N.: Semantic Web Technologies to Reconcile Privacy and Context Awareness. *Web Semantics Journal* **1** (2004)
5. Open Geospatial Consortium: OpenGIS Sensor Model Language (SensorML) Implementation Specification. OpenGIS Implementation Specification (Draft proposed version) 06C 05-086r2 (2006)
6. Wishart, R., Henricksen, K., Indulska, J.: Context obfuscation via ontological descriptions. In: Workshop on Location and Context Awareness (LoCA'05). Volume 3479 of Lecture Notes in Computer Science., Springer-Verlag (2005) 276–288
7. Lederer, S., Beckmann, C., Dey, A., Mankoff, J.: Managing Personal Information Disclosure in Ubiquitous Computing Environments. Technical Report IRB-TR-03-015, Intel Research Berkley (2003)
8. Open Geospatial Consortium: Observation and Measurements. OGC Best Practices Document (Draft) 06C 05-087r4 (2006)
9. Indulska, J., Henricksen, K., Hu, P.: Towards a standards-based autonomic context management system. In: Proceedings of the 3rd International Conference on Autonomic and Trusted Computing (ATC'06). Volume 4158 of Lecture Notes in Computer Science. (2006) 26–37
10. Henricksen, K.: A Framework for Context-Aware Pervasive Computing Applications. PhD thesis, University of Queensland, Information Technology and Electrical Engineering Department (2003)