Extending context models for privacy in pervasive computing environments*

Karen Henricksen[1], Ryan Wishart[2], Ted McFadden[1] and Jadwiga Indulska[2]
[1] CRC for Enterprise Distributed Systems Technology
{kmh, mcfadden}@dstc.edu.au
[2] School of Information Technology and Electrical Engineering,
The University of Queensland
{wishart, jaga}@itee.uq.edu.au

Abstract

Privacy is widely recognised as a significant obstacle inhibiting the adoption of context-aware applications. In order to remove this obstacle, advances are required in many areas of context-awareness research. In this paper, we address the incorporation of privacy support into context models. In particular, we present extensions to our context modelling approach that address the challenges of assigning ownership to context information and enabling users to express privacy preferences for their own information.

1. Motivation

Context-awareness is receiving increasing interest as a software design approach that is appropriate for pervasive computing. Context-aware software relies on various types of context information in order to make decisions about how to dynamically adapt to meet user requirements. This information is usually derived from a range of sources, including user profiles, applications, and sensors. Some types of context information are inherently sensitive and must be protected in order to satisfy users' privacy requirements.

Unfortunately, providing adequate protection for context information is extremely challenging. Context-aware systems typically contain collections of heterogeneous information with variable privacy requirements resulting from (i) differences in the sensitivity of the information, (ii) differences in users' individual privacy preferences, and (iii) changes in users' privacy preferences over time and in response to context changes. The problem of controlling access to context information is further complicated by the fact that pervasive computing environments permit looser and more dynamic couplings between people and resources, thereby invalidating the usual approaches to ownership and control of resources. In traditional computing systems, resources such as files are often assigned ownership according to who creates them, and owners control access to their resources by setting permissions. In contrast, in pervasive systems, there is often no direct link between an information source (e.g., a camera in a meeting room) and the entities that should be entitled to control the corresponding context information for privacy purposes (e.g., the people captured on camera). In addition, the relationship between these entities and the context attributes in which they have an interest in terms of privacy - which we refer to in this paper simply as an *ownership* relation - is often context-dependent.

As a result of these problems, most current context-aware applications provide very little support for privacy. Early efforts to address privacy concerns have investigated the design of privacy-preserving location sensing systems [10] and the integration of access control mechanisms into pervasive computing infrastructure [4, 2]. These solutions address only a small subset of the privacy challenges faced in context-aware systems, and are based on many simplifying assumptions (e.g., they only consider location information, or assume that context information is neatly partitioned into repositories that are under the control of a single user). In this paper, we tackle the ownership challenges described above in an attempt to provide one of the missing pieces required for a complete privacy solution for context-aware systems. We argue that ownership information forms a natural extension to context models, and propose the integration of flexible notions of ownership into our previously developed context modelling techniques. We also discuss some early results of our efforts to develop abstractions that users can exploit to express their privacy preferences.

2. Overview of our modelling approach

In the following sections, we briefly review our fact-based and situation-based approaches to context modelling.

*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).

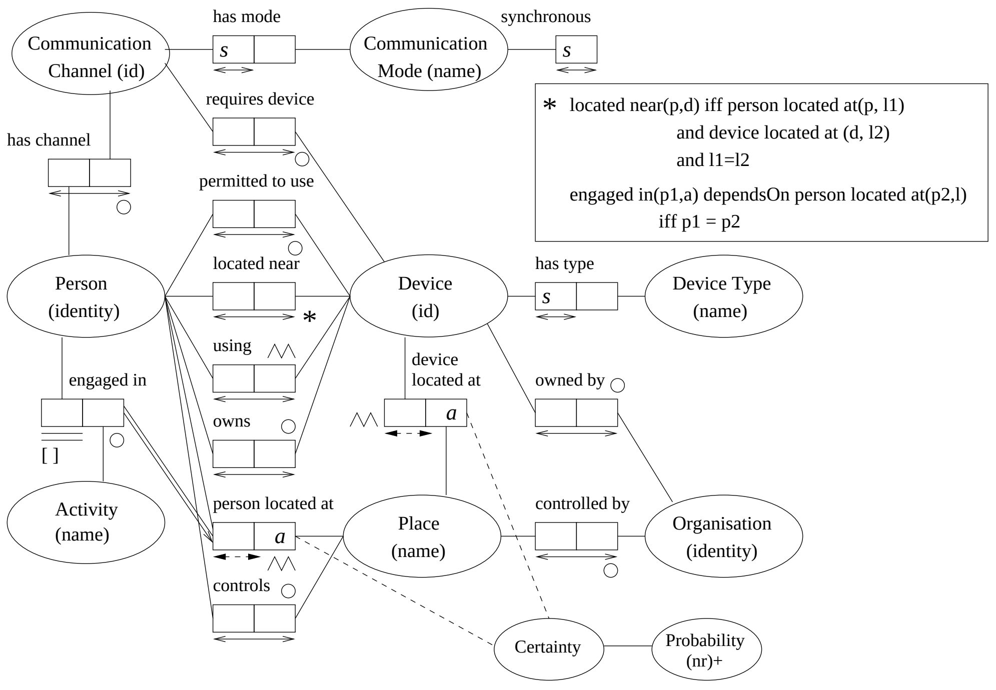

Figure 1. Modelling context for a communication application using CML.

2.1. Fact-based context modelling

Our fact-based modelling approach provides a tool that application developers can use to explore and formally specify the context requirements of a context-aware application. It provides constructs for defining the entities about which context information is required and the types of information (or facts) that are of interest in relation to each entity. It also allows developers to identify an appropriate source for each fact type, specify dependencies and constraints, and explore information quality issues. The two notations used to express the modelling constructs (graphical and textual) are outlined in the following sections.

2.1.1 Graphical modelling notation

As we have presented our graphical modelling notation in a previous paper [7], we provide only a brief summary here. The notation is based on Object-Role Modeling (ORM) [3]. ORM represents object types as ellipses and fact types, which are relations on one or more object types, as sequences of role boxes. Each object type is assigned a name and a reference mode that describes how instances of the type are represented. Fact types are annotated with uniqueness constraints, represented as arrows or lines spanning one or more role boxes; these place restrictions on the populations of the fact types in the manner of key constraints on attributes of relations in the relational model. A variety of other constraint types are also supported.

CML introduces various extensions to this basic notation, illustrated in an example model in Figure 1. The CML extensions allow fact types to be labelled as:

- *static* (s), *sensed* ($\wedge\wedge$), *derived* (*) or *profiled* ($\circ$) types, depending on persistence and source;
- *temporal* ([]) types that capture histories of context information; and
- *alternative* (a) types that are capable of describing ambiguous information (e.g., conflicting location reports gathered from a variety of location sensors).

CML also provides extensions to support special constraints on temporal and alternative fact types, annotation of fact types with appropriate metadata types, and dependencies between pairs of fact types.

2.1.2 Context schema notation

The context schema notation provides an alternative textual format for modelling context, based loosely on SQL. A context schema includes a declaration for each object and fact type, in which appropriate keywords are included to represent annotations on fact types. The schema notation offers the full expressiveness of the graphical modelling constructs, and incorporates additional detail that is inappropriate in a graphical model, including explicit naming of roles in fact types and full definitions of reference modes in terms of SQL data types. Context schemas provide a convenient input for tools that automatically generate helper

(a)

```
CREATE DOMAIN (
   Person AS Identity,
   Channel AS ChannelID,
   CommunicationMode AS ModeName...
)
```

(b)

```
CREATE STATIC FACT TYPE HasMode
(
   Channel channel KEY,
   CommunicationMode mode
)
CREATE TEMPORAL PROFILED FACT TYPE EngagedIn
(
   Person person SNAPSHOT KEY,
   Activity activity
)...
```

(c)

```
CREATE SITUATION CanUseChannel(person, channel):
    forall device
    . RequiresDevice[channel, device]
    . LocatedNear[person, device] and
      PermittedToUse[person, device]
```

Figure 2. Excerpts from a context schema representing the model shown in Figure 1.

classes that simplify the development of context-aware applications [9].

Context schemas are divided into several parts. The first part defines the object types and their representations, as shown in the example in Figure 2 (a). The second part describes the fact types in terms of previously defined object types, as illustrated in Figure 2 (b). Finally, the third part defines a number of situations based on the fact types of the second part. We discuss these in the following section.

2.2. Situation-based context modelling

Our situation abstraction provides a means to describe contexts in higher level terms than individual facts. Situations are defined using a variant of predicate logic, as shown by the example in Figure 2 (c), and can be easily combined using logical connectives to form increasingly rich context descriptions. They effectively express conditions on the context that can be evaluated against a set of variable bindings and a context (represented as a set of facts) to yield a truth value. Further information and examples can be found in an earlier paper [7].

3. Modelling ownership of context

In the remainder of the paper, we consider the problem of representing privacy requirements in relation to context information. Although our focus is on the context modelling approach outlined in the previous section, the concepts that we propose (ownership and context-dependent privacy preferences) can also be transferred into other modelling approaches.

This section proposes an ownership scheme that addresses the challenges introduced in Section 1 and is applicable at both the fact level and the situation level.

3.1. Modelling ownership of context facts

Context models such as the one presented in Figure 1 are often instantiated as large fact bases that merge information from a variety of sources about many different people and/or organisations. This implies that ownership must be captured using an approach that is scalable (in terms of specification of ownership relations and checking of ownership at run-time), yet sufficiently fine-grained to distribute control appropriately to both individuals and groups.

We propose a scheme in which ownership relations are specified at the level of the fact type, so that each type has its own rules for assigning ownership to individual facts. These rules associate each fact with zero, one or multiple owners. Facts that have zero owners are *public*; that is, they are not associated with any privacy preferences and can be freely disclosed to anyone. Non-public facts are always visible to their owners, but are disclosed to others only in accordance with the privacy preferences of all owners.

In order to make the task of specifying ownership of information more manageable, we also associate ownership with object types as described in Section 3.1.1. This removes the need to specify ownership on every fact type, as the default ownership of each fact type can be defined as the union of the ownerships of the objects participating in the fact type's roles. We have found that in most cases, this default ownership is actually the desired one. In the few cases where the default ownership is not appropriate, fact type ownership can be specified as described in Section 3.1.2.

3.1.1 Modelling ownership of object types.

As seen in our example context model, object types represent a variety of physical entities, such as people and devices, as well as abstract concepts, such as groups and classifications. Some of these object types have the capacity to act as owners of the context information related to them, while others do not. We refer to objects that are capable of ownership as *first class* objects. In our example model, there are two object types belonging to this class: `Person` and `Organisation`. For first class objects, ownership need not be explicitly specified. For each object type that is first class, we prepend the context schema declaration with the keywords "`FIRST CLASS`", as shown in Figure 3 (a).

Other objects, such as devices, communication channels and physical places, fall under the ownership of various first

(a)

```
FIRST CLASS Person AS Identity,
FIRST CLASS Organisation AS Identity,
```

(b)

```
SECOND CLASS Device AS DeviceID OWNED BY
   SELECT person FROM Owns
   WHERE Owns.device = Device
   UNION SELECT person FROM Using
   WHERE Using.device = Device
   UNION SELECT organisation FROM OwnedBy
   WHERE OwnedBy.device = Device,
SECOND CLASS Place AS PlaceName OWNED BY
   SELECT person FROM Controls
   WHERE Controls.place = Place
   UNION SELECT organisation FROM ControlledBy
   WHERE ControlledBy.place = Place,
```

(c)

```
THIRD CLASS CommunicationMode AS ModeName,
THIRD CLASS DeviceType AS TypeName,
```

(d)

```
CREATE SENSED ALTERNATIVE FACT TYPE PersonLocatedAt
QUALITY(Certainty)
(
   Person person KEY,
   Place place ALTROLE,
) OWNED BY person
```

(e)

```
CREATE SITUATION CanUseChannel(person, channel)...
OWNED BY person,
CREATE SITUATION Engaged(device)...
OWNED BY
   SELECT person FROM Owns
   WHERE Owns.device = device
   UNION SELECT organisation FROM OwnedBy
   WHERE OwnedBy.device = device
   UNION SELECT person FROM Using
   WHERE Using.device = device,
CREATE SITUATION WorkingHours()... UNOWNED
```

Figure 3. Adding ownership to the schema.

class objects. These are *second class* objects. The associations between first and second class objects can be context-dependent, and are stated explicitly in the context schema in terms of one or more fact types. Examples are shown in Figure 3 (b). Device information is owned by the owners and current users of the device, while place information is owned by the people and/or organisations that control the corresponding physical space.

Third class objects never have owners. In our example model, communication modes fall into this category, implying that all modes and all unary facts belonging to the `SynchronousMode` fact type are universally visible. Third class object types have simple schema declarations similar to those of first class types, as shown in Figure 3 (c).

3.1.2 Modelling ownership of fact types.

The default ownership derived from object ownership declarations can be overridden by defining a different ownership within a fact type declaration, as shown in Figure 3 (d). This example reassigns ownership of `PersonLocatedAt` facts to the individuals described by the facts, replacing the default scheme in which ownership is shared between these individuals and the entities that have ownership over places that appear in the facts. This modification gives individuals greater control over their location information. It implies that the people and/or organisations that control a place no longer have automatic access to knowledge about the people present in the place. Additionally, it ensures that a person's decision to disclose his/her location cannot be negated by a more restrictive policy belonging to the place owner.

3.2. Modelling ownership of situations

As discussed in Section 2.2, situations describe abstract classes of context defined in terms of fact types. One approach to handling privacy in relation to situations is to not specify ownership explicitly, but instead apply the rules associated with the fact types referenced by the situation. In this approach, the state of a situation is disclosed only when all of the facts upon which this state depends are also disclosed. However, this approach is unsatisfactory for two reasons. First, it is inefficient, as situations may depend on large numbers of facts, each having distinct owners and privacy preferences. Identifying all of the owners, retrieving the relevant preferences, and evaluating these for the current context is likely to be extremely time consuming. Second, it is unnecessarily restrictive. Queries on situations almost always reveal less information than those on facts, as they return simple truth values instead of precise values for attributes such as location. This means that a more permissive privacy policy can often be allowed. For these reasons, we assign each situation its own ownership as shown in Figure 3 (e).

The first example demonstrates that ownership of situations can be trivially linked to values of variables (the `person` variable, in this case). The second example illustrates context-dependent ownership: `Engaged(device)` is owned by a combination of the owners and users of the device. The final example shows a situation that has no owners and can be freely queried by anyone.

4. Modelling users' privacy requirements

Assigning appropriate ownership of context information is a first step towards supporting privacy. However, in order to implement privacy mechanisms such as access control, it is also necessary to provide a way for owners to stipulate the conditions under which their information can be disclosed to applications or users that request it. As in other privacy frameworks [8, 1], we represent these conditions in terms of privacy preferences. However, our model of privacy preferences differs from previously proposed models in that it offers all of the following features:

- inclusion of arbitrary types of context information in preferences, instead of no context information [1], or only limited types [8];
- specification of preferences over imperfect context information using advanced features of our context modelling approach [6]; and
- representations for both positive and negative preferences, so that users can describe circumstances in which information *can* and *cannot* be disclosed[1].

Our model for privacy preferences leverages the generic preference model that we described in [7]. This model captures preferences as pairs containing a scope and a scoring expression. The scope describes the context in which the preference is applicable in terms of previously defined situations and arbitrary variables, while the scoring expression assigns a score that indicates how appropriate an action described by the variable values is within this context.

Our model of privacy preferences is a restricted form of the general model, in which preferences are specified over the following variables:

- the *owner* of the preference;
- the person or application issuing the query (*requester*);
- the *purpose* for which the query results will be used;
- the *fact type* or *situation* that is being queried; and
- relevant fact type attributes or situation variables.

The scores we currently use are either *prohibit* (do not disclose) or *oblige* (do disclose), but we are also investigating scoring models that allow disclosure at different levels of granularity or precision. We are currently also developing a set of additional abstractions above our basic preference model to ensure that the task of creating complete profiles of privacy preferences is relatively straightforward for information owners. These will allow owners to dynamically define context classes in terms of arbitrary features of the context model (e.g., classifications of fact types), and then attach their preferences to these instead of directly to individual fact types and situations. We expect this feature to be very useful, as it is tolerant of an evolving context model and addresses observations made in previous research that privacy requirements are often different for static versus dynamic information [8], and for histories versus snapshots.

[1] As discussed by Agrawal et al. [1], preference specification is considerably simpler when both preference types are supported.

5. Future work

As well as refining our preference modelling approach, we are extending the work presented in this paper in other ways. One area we are investigating is how to better support disclosure of context information at different levels of detail as advocated by Hengartner and Steenkiste [5] and Lederer et al. [8]. Currently, we support this to some extent by allowing less restrictive privacy policies to be placed on situations than on the facts referenced by the situations. However, we are also investigating the use of obfuscation techniques to degrade the quality of selected values in facts according to users' preferences, and the arrangement of context information into hierarchies to facilitate information disclosure at different levels of granularity. Additionally, we are designing an access control mechanism to support privacy of users in pervasive computing environments, based on the concepts introduced in this paper.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *12th International Conference on World Wide Web*, Budapest, 2003.

[2] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal*, 1(3), 2004.

[3] T. A. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufman, San Francisco, 2001.

[4] U. Hengartner and P. Steenkiste. Access control to information in pervasive computing environments. In *9th Workshop on Hot Topics in Operating Systems*, Hawaii, May 2003.

[5] U. Hengartner and P. Steenkiste. Protecting access to people location information. In *1st International Conference on Security in Pervasive Computing (SPC)*, Lecture Notes in Computer Science, pages 25–38. Springer, 2004.

[6] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Workshop on Context Modeling and Reasoning (CoMoRea), 2nd IEEE Conference on Pervasive Computing and Communications (PerCom)*, pages 33–37, Orlando, March 2004.

[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)*, Orlando, March 2004.

[8] S. Lederer, C. Beckman, A. Dey, and J. Mankoff. Managing personal information disclosure in ubiquitous computing environments. Technical Report IRB-TR-03-015, Intel Research, Berkeley, June 2003.

[9] T. McFadden, K. Henricksen, and J. Indulska. Automating context-aware application development. In *UbiComp Workshop on Advanced Context Modelling, Reasoning and Management*, Nottingham, September 2004.

[10] G. Myles, A. Friday, and N. Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, January-March 2003.