An Architecture for Privacy-Sensitive Ubiquitous Computing

ABSTRACT

Privacy is the most often-cited criticism of ubiquitous computing, and may be the greatest barrier to its long-term success. However, developers currently have little support in designing software architectures and in creating useful and usable interactions that are effective in helping end-users manage their privacy. To address this problem, we present Context Fabric, an infrastructure for facilitating the development of privacy-sensitive ubiquitous computing applications. The requirements for Context Fabric were gathered through an analysis of end-user needs and application developer needs for privacy. Context Fabric provides basic support for building ubiquitous computing applications, providing several customizable privacy mechanisms as well as a framework for extending privacy functionality. These mechanisms facilitate the creation of three basic interaction patterns for privacy-sensitive applications: optimistic, pessimistic, and mixed-initiative. Combined, these features allow application developers and end-users to support a spectrum of trust levels and privacy needs.

1. INTRODUCTION

Westin defined information privacy as "the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [65]. While many people believe that ubiquitous computing holds great promise, privacy is easily its most often-cited criticism. There have been numerous interviews (e.g. [7, 33, 40]), essays (e.g. [20, 62, 64]), books (e.g. [10, 27]), and repeated negative media coverage (e.g. [59, 66]) describing people's concerns about the strong potential for abuse, general unease over a potential lack of control, and desire for privacy-sensitive ubicomp systems. These concerns suggest that privacy may be the greatest barrier to the long-term success of ubiquitous computing.

The large majority of previous work on privacy has tended to focus on providing anonymity or on keeping personal information and messages secret from hackers, governments, and faceless corporations. While anonymity and secrecy are important, they only address a relatively narrow aspect of privacy and do not cover the many situations in everyday life where people *do* want to share information with others. For example, one could imagine sharing one's location information with friends to facilitate micro-coordination of arrivals at a meeting place, or sharing simple notions of activity to convey a sense of presence to co-workers and friends. It is important to note here that the parties that are receiving such information already know one's identity, are not adversaries in the traditional sense, and that the privacy risks may be as simple as wanting to avoid undesired social obligations or potentially embarrassing situations.

The point is that, rather than being a single monolithic concept, privacy is a fluid and malleable notion with a range of needs and trust levels. *The key here is in empowering people with choice and informed consent, letting individuals share personal information with the right people and services, in the right situations, and at the right level of detail.* As Weiser noted, "The problem, while often couched in terms of privacy, is really one of control. If the computational system is invisible as well as extensive, it becomes hard to know what is controlling what, what is connected to what, where information is flowing, how it is being used…and what are the consequences of any given action" [64].

However, the problem is that it is still difficult to build privacy-sensitive ubicomp applications. Previous work, such as the PARCTab system [57], the Context Toolkit [19], and iROS [39], has looked at providing general support for building ubicomp applications, but these systems do not provide features for managing privacy. Consequently, system developers have little guidance or programming support in creating architectures and user interfaces that are effective in helping end-users manage their privacy. The result is that privacy is done in an ad hoc manner and often as an afterthought, if at all, leading to applications that end-users may ultimately reject because they are uncomfortable using them or find them intrusive.

Based on an analysis we performed of end-user privacy needs and application developer needs, we are developing Context Fabric (Confab), an infrastructure aimed at simplifying the task of creating privacy-sensitive ubicomp applications. Confab is tailored for context-aware computing [58], a common aspect of ubiquitous computing in which sensors and other data sources are leveraged to provide computing systems with an increased awareness of a user's physical and social environment. From a software architecture perspective, Confab provides an architecture and a suite of privacy mechanisms that allow application developers and end-users to support a spectrum of trust levels and privacy needs. From an end-user perspective, Confab facilitates the creation of three basic interaction patterns for privacy-sensitive applications: *optimistic*, where an application shares personal information and detects abuses by default; pessimistic, where it is more important for an application to prevent abuses; and mixed-initiative, where decisions to share information are made interactively by end-users.

It should be noted that Confab is not intended to provide perfect privacy, if there is even such a thing. There are many social and organizational issues that simply cannot be managed by technological means alone. Ultimately, privacy will have to be managed through a combination of technology, legislation, corporate policy, and social norms. What Confab provides is an overall architecture and an extendable suite of techniques to make it easier for developers to build privacy-sensitive applications for an intended community of users, and for companies to offer their ervices while minimizing the risk to people's privacy. As an analogy, a web design tool can be used to create good as well as bad web sites, but a useful tool will be oriented to make it easier to create good ones.

The rest of this paper is organized as follows. First, we examine the requirements for privacy-sensitive infrastructure. We continue with a description of Confab's architecture and how it supports those requirements. Then, we describe our evaluation of Confab with two applications we have built using it. We wrap up with a comparison to related work, future work, and conclusions.

2. SYSTEM REQUIREMENTS

The primary metric of success for any infrastructure is if it can be used to create a useful and non-trivial subset of the full design space of applications in a manner that is faster, is higher quality, or has more useful features than without it. In this section, we explore the targeted design space, looking at the requirements we gathered by analyzing end-user needs as well as application developer needs.

2.1 End-User Needs

The end-user needs for Confab were gathered through several scenario-based interviews on location-enhanced applications we performed with twenty people of various ages, professions, and levels of computer expertise; extended analysis of freeform comments on a previous survey on ubicomp privacy preferences [46]; analysis of research papers [7, 13, 31, 33, 35, 36, 40, 46] and message boards [1] on hypothetical and actual usage of emerging ubicomp systems where privacy was an issue; analysis of several proposed and existing privacy protection laws [2, 21, 26]; and analysis of several different design guidelines for privacy-sensitive systems [4, 8, 51], in particular the fair information practices [45, 65] and asymmetric information flows [38]. A full discussion of the results and implications of our analysis are beyond the scope of this paper. Instead, we provide a summary of six major themes below.

First, people are concerned about systems that centralize data [7, 36]. While there are many advantages to centralized architectures, it also means that sensitive data is stored on a computer that end-users have little practical control over. For example, Hong et al [36] note that while a visible effort was made to create written privacy policies about how location information was used in the PARCTab system [57], users still had the perception that if the research team managing the system changed their policies, or if upper-level managers wanted to examine the data, there was little they could do about it. Similar debates have emerged over the deployment of E911 in the United States. Another drawback is that centralized servers are attractive targets for computer security attacks.

Second, people want simple and appropriate levels of control over who sees what information about them and when [1, 13, 33, 35, 36]. Each person has different levels of trust in their relationships with other people and organizations. For example, in our surveys and interviews, many people said that they would be willing to share their location information with their friends and families, but had some concerns about sharing this information with co-workers, and especially so with companies that did not offer tangible value and adequate privacy protection. People also expressed concerns about continuous flows of information versus discrete flows. For example, many of our interviewees said they would be comfortable with co-workers getting snapshots of their current location, but would be less comfortable if co-workers could continuously get their location information since

that could be more easily used to monitor their whereabouts.

Third, people want simple and appropriate feedback about what personal information is being disclosed. For example, the PARCTab system provided no feedback about what information was revealed to third parties [36, 57]. A stranger could monitor a user's location by making repeated queries about the user's location without that user knowing. Previous research has focused on flexible, yet complex access control mechanisms; however, our surveys and interviews suggest that in many cases, simple access control and basic notifications are sufficient. Access control can be used to select who can view personal information, with notifications providing social visibility to prevent abuses. For example, Alice is less likely to repeatedly query Bob's location if she knows that Bob can see each of her requests. This insight significantly influenced the design of our architecture, and is discussed in further detail in the next subsection.

Fourth, many people expressed a desire for plausible deniability. Our survey and interviews, as well as Hindus et al's previous work on ubiquitous computing in the home [35] have suggested a social need to avoid potentially embarrassing situations, undesired intrusions, and unwanted social obligations. For example, it is not uncommon for an individual to answer with a white lie when asked on the phone where they are or what they are doing. Cell phones are a good example of a system that provides plausible deniability, in that if a person does not answer a call, it could be for technical reasons—such as being outside of a cell, not having the phone with them, or that the phone is off—or for social reasons, such as being busy or not wanting to talk to the caller right now. By default, it does "the right thing."

Fifth, some of our interviewees were concerned over long-term retention of personal information, as it opens up the possibility for data mining. As Grudin notes [31], one of the problems with capturing physical world information in digital formats is that it "can show up any place, anytime. It may not, but it could. So we never truly know the full context in which we act—where, when, or by whom our actions may be interpreted." As a current example, it is not uncommon for people to use a search engine to find information about a prospective employee, taking away from that individual's control over disclosure.

Sixth, people expressed that there should be special exceptions for emergencies. In these situations, safety far outweighs any privacy needs. In our interviews, people noted that E911 made sense if it transmitted location information only when making the call, but should not transmit information at any other times. Cadiz and Gupta also note that trusted proxies are sometimes used to handle these kinds of situations [13]. For example, MedicAlert is a paid service that stores personal medical records and sends it in the case of medical emergencies.

Table 1 shows a summary of the end-user requirements.

End-user requirements
• Decentralized architecture • Simple and appropriate control • Simple and appropriate feedback • Plausible deniability • Limited retention of data • Special exceptions for emergencies

Table 1. Summary of end-user requirements.

2.2 Application Developer Needs

The application developer needs for Confab were gathered by identifying privacy functions common in several networked as well as ubicomp applications. We examined research prototypes and emerging commercial applications, limiting the scope to what we call personal ubiquitous computing, that is systems where data starts with the end-user and can optionally be disclosed to others in a limited manner. We also chose to focus more on location than on other forms of contextual information, since a sizeable number of this type of application is emerging in the market, and thus has a clearer path to widespread use. We were also influenced by the Geopriv requirements for location privacy [17] and Jiang et al's work on asymmetric information flows [38].

The genres of applications we have identified include messaging systems, such as cell phones, instant messenger, SMS, and messaging within a home [48] and between homes [35]; guides for exploration and navigation [3, 50]; finders for finding people, places, or things [6, 30]; group awareness displays [19, 30]; augmented-reality games [24, 28]; contextual information tagging and retrieval, including personal memory aids [11, 43, 55], associating topical information with places [12, 23, 52, 58]; situational real-time information (such as local weather or traffic); and enhanced safety for individuals and emergency responders [25, 47].

From a systems standpoint, there are several basic features that need to be supported, including acquiring context data from a variety of sources, refining and storing that context data, and retrieving and using context data. This last issue, retrieving and using, can be done either through push transactions (e.g., you

sending your location information during an E911 call) or through pull transactions (e.g., a friend requesting your location). For each of these types, there is also a need for continuous sharing, where personal data is constantly forwarded to another party (e.g., allowing a health monitoring system to repeatedly forward data to your doctor), as well as for discrete disclosures that happen intermittently or one time only. These are basic features that are mostly supported by other systems aiding the development of ubicomp applications (e.g. [19, 57]).

From a privacy standpoint, we have identified six common features that need to be supported. The first is support for three basic interaction patterns for privacy-sensitive applications: pessimistic, optimistic, and mixed-initiative. In *pessimistic* applications, end-users set up preferences beforehand, placing strict requirements on when personal information can flow to others. In contrast, *optimistic* applications [53] are based on the maxim that it is easier to ask for forgiveness than it is to ask for permission. In it, end-users generally allow greater access to personal information but try to detect abuses after the fact with logs and notifications. For example, AT&T mMode's Find Friends [6] lets end-users set up a list of people that can request one's location. End-users then receive notifications when their location is requested. Optimistic access control is useful in cases where openness and availability are more important than complete protection. Optimistic access control also has the advantage that it is far easier for people to use, since it is rather difficult for individuals to predict all of the possible usage scenarios (and thus all of the necessary permissions). In *mixed-initiative* control, end-users are interrupted when someone requests their personal information and must make a decision then and there. An example is choosing whether or not to answer a phone call given the identity of the caller.

The second feature is granular control governing the level of precision and the rate of disclosures. As an example of precision, one could choose to disclose one's location as "123 Montgomery Street" or "San Francisco", or one's activity as "writing a paper" or "busy" depending on who is requesting the information and on the situation. As an example of rate of disclosure, the application could be designed to allow access every 15 seconds or every hour. The simplest example of this is invisible mode, a common feature in messenger clients where no information is disclosed.

The third is limited data retention, providing support for automatically deleting old data. This is something espoused by the European Directive [2] and the proposed Location Privacy Act of 2001 [21]. For example, web browsers do this with their history of previously viewed pages, as do some cell phones with their call history. Limited data retention also has the advantage of limiting the damage that is done should an accidental or malicious leak occur.

The fourth is support for tagging personal information as it flows to others, as described by both Geopriv [17] and by Korba and Kenny [42]. The idea here is that such information can be marked with personal preferences about, for example, whether it should be forwarded to others or how long it should be retained before deleting. These tags can also be used as a fingerprint on the data to help with tracking and auditing as well. This is an approach taken by many Digital Rights Management systems.

The fifth is fine-grained mechanisms for controlling the access, flow, and retention of personal information. These mechanisms include restrictions based on identity, location (e.g., only allow inquirers in the same building as me to see my location), and time (e.g., let my co-workers see my location between 9AM and 5PM).

The sixth is logs, both for clients and servers. On the client side, logs that are summarized in a compact form make it easier for end-users to understand who is accessing what data. On the server side, logs make it easier for service providers to audit their activities to ensure that they are handling their customers' personal information properly. On both sides, logs also make it possible to apply machine learning techniques to detect unusual access patterns that might indicate abuses of someone's personal information.

Table 2 shows a summary of the application developer requirements.

Application Developer requirements
• Support for optimistic, pessimistic, and mixed-initiative applications • Granular control over precision and rate of disclosure • Limited retention of data • Tagging of personal information • Fine-grained mechanisms for access, flow, and retention • Logging

Table 2. Summary of application developer requirements.

2.3 Summary of Requirements

We have organized the end-user and application developer requirements into four high-level requirements below.

- A decentralized architecture, where as much personal information about an end-user is captured, stored, and processed on local devices owned by that end-user
- A range of mechanisms for control and feedback by end-users over the access, flow, and retention of personal information, to support the development of pessimistic, optimistic, and mixed-initiative applications.
- A level of plausible deniability built in
- Special exceptions for emergencies

It is important to note here that the requirements presented above are intended to support a range of privacy policies rather than all being used in a single application. As described by Jiang et al [38], different communities of users have different trust relationships. Thus, there is a spectrum of privacy needs, and ubiquitous computing applications should be tailored to those needs.

3. CONFAB SYSTEM ARCHITECTURE

In this section, we describe the architecture for Confab, describing its data model and programming model.

In designing and developing Confab, we had two requirements in addition to the ones listed in the previous section: making the architecture and applications easy to understand by end-users, and easy to use by programmers. Given the choice between a simpler solution and a more powerful but complex one, we usually opted for the simpler one, to make it easier for end-users to understand what data the system knew about them and where that data was flowing, and to facilitate adoption by application developers.

We focus primarily on systems where personal information about an end-user starts with that end-user. We believe that there will be a useful and non-trivial subset of ubicomp applications that are built along these lines, for two reasons. First, over the past few years, the research community has been moving from centralized location-tracking architectures (e.g., [63]) to decentralized location-support ones (e.g., [54]) for reasons of scalability and privacy. We believe that future research will make it possible to provide privacy protections in the physical layer for other forms of personal contextual information. Second, there is already a large market for personal items in which sensors can be cheaply embedded, for example PDAs, home security systems, and cars. Although Confab could be used in cases where data is initially captured by others (e.g., smart rooms or surveillance cameras), we do not explicitly address those cases.

3.1 Usage Scenario

In this section, we describe two scenarios to help illustrate what kinds of applications we want to support and how they would work within Confab.

Scenario 1 – Find Friend

Alice's workplace has setup a new server that employees can use to share their location information with one another. Employees can choose to share their location information by uploading updates to the server at the level they desire, for example at the room level, at the floor level, or just "in" or "out". To help allay privacy concerns, the server is also set up to provide notifications to a person whenever their location is queried, and to accept queries only if the requestor is physically in the same building.

Scenario 2 – Mobile Tour Guide

Alice is visiting Boston for the first time, and wants to know more about the local area. She already owns a location-enabled device, so all she needs to do is find a service that offers an interactive location-enhanced tour guide and link her device to it. She searches online and finds a service named Bob that offers such tour guides for a number of major cities. She decides to download it and try it out.

When starting the application, Alice discovers that Bob offers three levels of service. If Alice chooses to share her location at the city level, Bob can tell her how long the lines are at major venues such as museums, and what calendar events there are. If she shares her location at the neighborhood level, Bob can tell her what interesting shops there are and nearby points of interest. If she shares it at the street level, Bob can offer her real-time maps and a route finder that can help her navigate. The application also states that Bob will retain her location data for up to 3 months, and at the neighborhood level sends updates of her location to Bob every 10 minutes when the application is running.

Since this is her first time using the service, and since she has not heard of Bob before, Alice decides to share her location information at the neighborhood level.

3.2 Data Model

The goal of Confab's data model is to represent contextual information, such as one's location or activity. People, places, things, and services (entities) are assigned *infospaces*, network-addressable logical storage units that store context data about those entities (see Figure 1). For example, a person's infospace might have static information, such as their name and email address, as well as dynamic information, such as their location and activity.

Sources of context data, such as sensors, can populate infospaces to make their data available for use and retrieval. Applications retrieve and manipulate infospace data to accomplish context-aware tasks. Infospaces also provide an abstraction with which to model and control access to context data about an entity. For example, individuals can specify privacy preferences for how their infospace handles access control and flow (described in greater detail below).

Infospaces are managed by *infospace servers*, which can be either distributed across a network or managed centrally, analogous to how a person could choose to have their personal web site hosted on their home machine or by an ISP. Here, we focus on the case where infospaces represent contextual information about individuals, and are hosted on devices owned by those individuals.

The basic unit of storage in an infospace is the *context tuple*. Tuples are used to represent intrinsic and extrinsic context, as well as static and dynamic contextual information (see Table 3). Attributes of interest common to all tuples are *datatype*, a textual name describing the relationship of a tuple to the containing infospace's entity (for example, location or activity); *dataformat*, a string that describes the meaning of the data (for example, temperature could be Farenheit or Celsius); an optional *entity-link* denoting the address of an infospace for an entity described by the tuple; and one or more *values*, each identified by name (see Figure 2 for an example). Infospaces can store tuples containing arbitrary data, many of which may describe other entities related to the original infospace. Such tuples' entity-link attributes refer to the infospace of the other entity. For instance, the infospace for a specific room may contain numerous tuples of type 'occupant', each with values denoting a name and email of an occupant of the room and an entity-link referring to the infospace that hold tuples on behalf of that occupant.

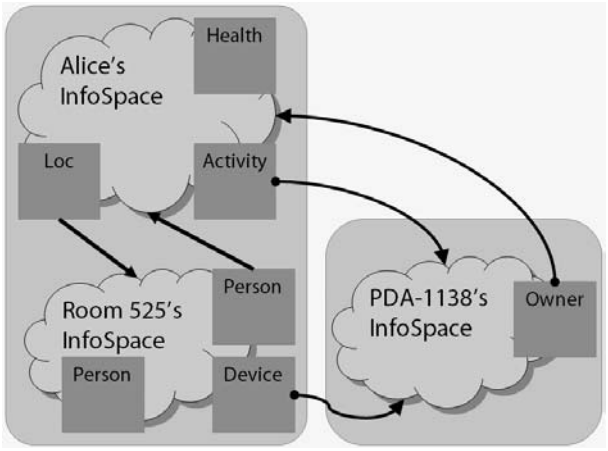

Figure 1. An infospace (represented by clouds) contains contextual data about a person, place, or thing. Infospaces contain tuples (squares) that describe individual pieces of contextual data, for example Alice's location or PDA-1138's owner. Infospaces are contained by Infospace servers (rounded rectangles).

Each tuple can also optionally have a privacy tag which provides end-user hints on how the data should be used when it flows to other infospaces. The current implementation of privacy tags provides hints on when a tuple should be deleted, to help enforce limited data retention. End-users can have their tuples tagged with a

	Intrinsic	Extrinsic
Static	Name, age, email address	A room is part of a building
Dynamic	Activity, temperature	A person is in a specific room

Table 3. Confab supports different kinds of context data. Static context data does not change or changes very slowly, whereas dynamic context data changes often. Intrinsic context data represents information about that entity itself, whereas extrinsic context data represents information about an entity in relationship to another entity.

```
<ContextTuple dataformat="edu.school.building"
              datatype="location"
           description="location of an entity"
           entity-link="http://myhost.com/~jdoe"
           entity-name="John Doe"
     timestamp-created="2003.Feb.13 16:06 PST">

   <Values>
      <Value value="523" />
   </Values>

   <Sources>
      <Source datatype="location"
                 link="http://localhost/map.jsp"
               source="Location Simulator"
            timestamp="2003.Feb.13 16:06 PST"
                value="523" />
   </Sources>

   <PrivacyTags>
      <Notify value="mailto:addr@mail.net" />
      <TimeToLive value="1 day" />
      <MaxNumSightings value="5" />
      <GarbageCollect>
         <Where requestor-location=
                  "not edu.school.building " />
      </GarbageCollect>
   </PrivacyTags>

</ContextTuple>
```

Figure 2. An example tuple. Tuples contain metadata describing the tuple (e.g., dataformat and datatype), one or more values, one or more sources describing the history of the data and how it was transformed, and an optional privacy tag that describes an end-user's privacy preferences. In this example, the privacy tag specifies a notification address, a maximum time to live, the maximum number of past values that should be retained, and an additional request to delete the data if the requestor is not in the specified location.

TimeToLive, which specifies how long data should be retained before being deleted; *MaxNumSightings*, which specifies the maximum number of previous values that should be retained (for example, a value of 5 means only retain the last five places I was at); *Notify*, which specifies an address to send notifications of second use to; and *GarbageCollect*, which specifies additional hints on when the data should be deleted, for example, when the current holder of the tuple has left the area.

By default, when a tuple of any datatype is requested, its value is "UNKNOWN", regardless of whether it actually exists or not. Requests can see correct tuple values only if they have been granted access. This approach provides some level of plausible deniability, as a datatype might be unknown due to technical failures, lack of actual data, restricted access, or because the person is in invisible mode.

Infospace servers, infospaces, and context tuples are currently implemented using standard web technologies. Infospace servers are built on top of web servers, simplifying deployment and providing a clear mental model for programmers and end-users. Individual infospaces are named via URLs, and can be thought of as web-based tuplespaces with specialized constraints. Context tuples are represented externally as data-centric XML documents.

3.3 Programming Model

From a high-level perspective, Confab supports a hybrid blackboard and dataflow architecture. Personal information is stored in infospaces that are running either in clients, proxies, and servers, with data flowing between these components in a controlled fashion. In this section, we describe how developers can make use of three different pieces—operators, active properties, and service descriptions—to build applications.

Methods and Operators

Each infospace supports two general kinds of methods, *in* and *out*. In-methods affect what data is stored within an infospace, and include add and remove. Out-methods govern any data leaving an infospace, and include query, subscribe, unsubscribe, and notify.

Each infospace also contains operators, pieces of chainable code, for manipulating tuples. Operators can be added to an existing infospace to extend and customize it to what is needed. Confab supports three different kinds of operators: in, on, and out. *In-operators* are run on all tuples coming in through in-methods. An example in-operator is one that checks the infospace's access control policies to make sure that this is a tuple that is allowed to be added. *Out-operators* are run on all tuples going out through out-methods. An example out-operator is one that blocks all outgoing tuples if the user is in invisible mode. On-operators are operators that run periodically, such as garbage collection. Table 4 shows a full list of operators provided in Confab by default.

Operator Type	Description
In	Enforce access policies Enforce privacy tags Notify on incoming data
Out	Enforce access policies Enforce privacy tags Notify on outgoing data Invisible mode Add privacy tag Interactive
On	Garbage collector Periodic report

Table 4. Confab provides several operators by default. Operators can be added or removed to customize what personal information a tuple contains and how it flows to others.

The Enforce Access Policies operators let end-users specify access policies for their infospace. Several different conditions can be specified for authorization, including who is requesting the data, what data they are requesting, how old the data is, what domain they are requesting from, and the current time.

The Enforce Privacy Tags operators are used to put the preferences specified in privacy tags into action. The out-operator version makes sure that data that should not leave an infospace does not, while the in-operator version does the same with incoming data. Together, a set of infospaces can provide peer enforcement of privacy tags, helping to ensure that data is managed properly (see Figure 3). Assuming that tuples are digitally signed, peers can also detect if privacy tags have been altered, thus detecting that an infospace is not handling personal information properly.

The Notify operators are used to send short messages to give end-users feedback about who is requesting information and when. Notify operators can be configured to send messages either through email or via instant messenger.

The Invisible mode operator can be used to block all outgoing tuples, returning a single value of "UNKNOWN". The Invisible mode operator can also be configured to return some pre-specified value. The Add Privacy Tag operator is used to add end-user or application defined privacy tags on outgoing tuples.

The Interactive operator can be used to give end-users control over disclosures. In the current implementation,

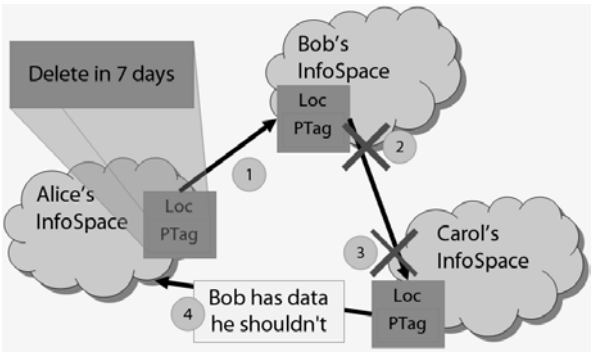

Figure 3. An example of peer enforcement. (1) Alice shares her location data with Bob. This data has been tagged to be deleted in seven days. Suppose seven days have passed, and that Bob passes the data on to Carol. If this is an accidental disclosure, then (2) his infospace prevents this from occurring. If this is intentional, then (3) Carol can detect that Bob has passed on data that he should not have, and (4) notifies Alice.

when a request comes in and the Interactive operator is active, a short message is sent, giving the end-user several options, including disclosing the requested information, degrading its quality, returning unknown, or granting access always or temporarily.

The Garbage Collector operator is run periodically to delete data which have privacy tags that specify that they should be deleted. The Periodic Report sends an email to the owner of an infospace, providing a summary of who has requested what.

Operators are added to an infospace in linear order, which dictates their order of execution. Each operator also has a filter that checks whether or not it should be run on a specific tuple. When an in- or out-method is called, a chain of the appropriate operators is assembled and then run on the set of incoming or outgoing tuples.

Note that peer enforcement and automatic deletion of old data can be trusted to execute on computers that an end-user has control over, but not necessarily on computers owned by others. Short of a trusted computing base, there is no way of forcing others to delete data. Privacy tags let end-users provide a hint saying what their privacy preferences are, and relies on social, legal, and market mechanisms that others will do the right thing. In cases where there is a strong level of trust, this will suffice and can help prevent accidental disclosures. In cases where there is not a great deal of trust, other mechanisms, such as passing on coarser-grained data or anonymity, should be used.

Active Properties

To simplify the task of maintaining context state in applications, Confab also provides an *active properties* object in which queries and subscriptions can be placed and values easily retrieved (see Figure 4).

Active properties supports three different kinds of properties: OnDemandQuery, which makes a request for new data whenever its value is checked; PeriodicQuery, which periodically checks for new data; and Subscription, which periodically receives new data from an infospace. After initial setup, clients can simply request the property name (e.g., "alice.location") to retrieve the last-known value.

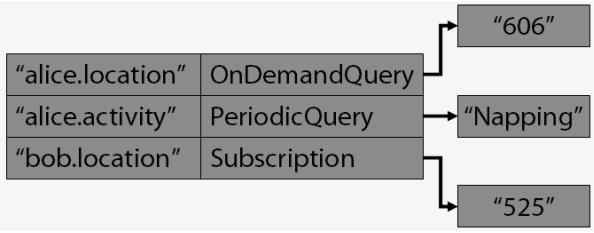

Figure 4. Clients can maintain a list of properties they are interested in through an active properties object.

Service Descriptions

Servers can publish services that describe various options that end-users can choose from. For example, Scenario 2 described a mobile tour guide service that offered different kinds of information depending on the precision of information Alice was willing to share.

Context Fabric provides support for servers to specify these different options, as shown in Figure 5. These service descriptions provide information about what the option offers, what datatypes and dataformats are needed, a hint about how often the information should be pushed to the server, and what URL the data should be pushed to. There are no built-in discovery mechanisms for service descriptions, however. We assume that there will be other mechanisms, for example advertising in magazines, a link on a web page, or finding it through search engines.

In summary, Confab's data model and programming model provides application developers with an architecture and a suite of mechanisms for building privacy-sensitive applications. Combined, developers have greater choice about where personal information is stored and managed, where it is processed, and what control and feedback end-users have about the flow of their personal information between actors.

3.4 Implementation

Confab is implemented in Java 2 v1.4.2, and is currently comprised of 450 classes and approximately 23,000 lines of code (not counting comments, blank lines, and boilerplate). Confab uses HTTP for network communication, and is built on top of the Tomcat web server, making extensive use of Java servlets. XPath is

```
<Service admin-email="bob@bob.com"
        description="A tour guide for cities"
           keywords="tourism"
               name="Bob's mobile tour guide"
           provider="Bob Inc"
    service-webpage="http://bob.com/tour/">

  <Option dataformat="location.gps"
           datatype="location"
              delay="15 seconds"
         description="Real-time maps, routes"
           infospace="http://bob.com?k=f45"
    />

  <Option dataformat="location.city"
           datatype="location"
              delay="20 minutes"
         description="museum lines, calendar"
           infospace="http://bob.com?k=f45"
    />

</Service>
```

Figure 5. Confab's service descriptions allow services to give end-users various choices when using a service. This example shows the service description for a mobile tour guide service. The first option lets end-users push their location information to the specified URL at the GPS level, with a hint that it should be sent every 15 seconds. In return, they will receive real-time maps and a route finder service.

used as the query language for matching and retrieving XML tuples.

Confab also comes with a microphone source, which is used to estimate activity level, and several web-page based simulators for faking location and activity data. We are currently in the process of hooking up live sensors to Confab.

4. EVALUATION

In this section, we describe the implementation of applications we have built on top of Context Fabric.

Find Friend

Using Context Fabric, we have built portions of the Find Friend application into a custom instant messenger client. Rather than a centralized server, our current implementation sends out periodic queries every 30 minutes to each person's infospace. Figure 6 shows the current user interface of this tool. At the top is information about the current user, in this case Alice. This provides feedback about what the system currently knows about her. On the bottom is a set of co-workers and a set of friends. End-users can also use this user interface to set up privacy preferences, specifying simple rules of what information each group should see, such as co-workers only seeing location information during regular work hours.

Our implementation currently consists of 1100 lines of code across 12 classes, most of which handle GUI interactions. It uses the Jabber client to connect across different instant messenging systems.

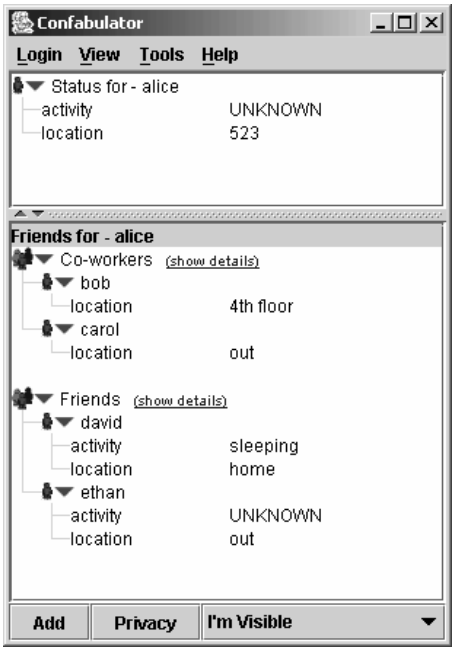

Figure 6. This is an instant messenging client that also shows location and activity information. This interface also lets end-users customize what information each group can see about them.

Emergency Response

We have also built an emergency response service that lets end-users share their location information with a building's emergency response infospace. People can pass in a link pointing to a trusted third party, such as a spouse or a paid service. This third party continually gets updates of the person's location. Thus, if there is ever an emergency, say for example a fire or earthquake, firefighters can issue a query to all of the links in a building and get the location information, with the third parties being notified that the location information has been disclosed. This approach allows emergency responders to get critical location information, provides a level of redundancy should the user's device or location systems fail, and provides a basic level of privacy.

We have also used Context Fabric to build prototypes of applications that have minimal privacy concerns. One that is currently in progress is emergency response support to help firefighters on scene. The prototype uses sensors and PDAs to automatically gather and disseminate information about the fire and that firefighter to other nearby firefighters. Our early evaluation of this system with firefighters has been promising. Another is a distributed querying system for supporting database operations, such as join or project, for streaming data and across multiple infospaces [34].

5. RELATED WORK

There has also been a great deal of work at providing programming support for various aspects of ubiquitous context-aware computing. This includes the PARCTab system [57], Cooltown [41], the Context Toolkit [19], Contextors [16], Limbo [18], Sentient Computing [5], Stick-E notes [52], MUSE [14], SpeakEasy [22], Solar [15], XWeb [49], GAIA [56], one.world [29], and iRoom [39]. Confab shares many characteristics with much of this work, and builds on it by focusing the architecture on privacy and on providing flexible mechanisms for privacy protection.

Confab is closest in terms of data model and programming model to the PARCTab system [57] and iRoom [39]. In many ways, Confab's data model can be thought of as a logical evolution of the PARCTab's Dynamic Environments. Dynamic Environments were centralized data stores associated with relatively large places, such as buildings. Each Dynamic Environment contained personal information about each people, places, and things within its purview. As people moved from place to place, they would also switch which Dynamic Environment they were using. The key differences Confab makes are greater decentralization of data, a greater range of mechanisms for privacy in both the data model and in the programming model, and compartmentalized extensibility thru operators. These mechanisms support the development of optimistic and mixed-initiative interfaces.

The iRoom is an interactive workspace supported by a suite of software. Central to this is the EventHeap, a shared tuplespace for the room in which input devices can place events and output devices can receive events. This level of indirection encourages looser coupling between application components and fosters greater overall robustness. Confab uses a similar approach with its infospaces, separating sources of data (such as sensors) from the services and applications that use them, with little or no knowledge of each other. Like the EventHeap, Confab also has a thin API with few methods. The main difference between the EventHeap and Confab is that Confab is specialized for building privacy-sensitive systems. Confab also looks at supporting multiple infospaces to represent people, places, and things, rather than just one tuplespace to represent events within a place.

There has also been some previous work on using digital rights management in managing personal information. Langheinrich [44] described pawS, a privacy awareness system for ubicomp that lets deployed systems announce P3P policies of what data is being collected, and database support for enforcing those policies. Similarly, IBM has also introduced an Enterprise Privacy Authorization Language [37] which lets developers describe privacy policies and attach those privacy policies to the data as it flows through a company. The privacy tags in Context Fabric are similar in spirit to these ideas, and introduces further digital rights management ideas, such as using location as a parameter, a maximum number of past sightings, and peer enforcement.

Context Fabric also builds on the work by Spreitzer and Theimer [60], who describe an architecture for providing location information. In their architecture, each user owns a User Agent that collects and controls all personal information pertaining to its user, and any request for such information must be routed through the User Agent which enforces predetermined access policies. Confab takes this same basic approach and extends it with a wider range of privacy mechanisms, including notifications, tags, logging, and interactive requests, to support the development of pessimistic, optimistic, and mixed-initiative type applications.

There has been a great deal of work in providing levels of anonymity in networked systems. One system of note here is Gruteser and Grunwald's work on spatial and temporal cloaking [32], in which a trusted proxy is used to adjusting the resolution of location reported to services based on the density of users in a region. Since many users report their location through the proxy, user density is known. Thus, the proxy can provide k-anonymity, that is hiding one's precise location by returning an area that has k-1 other people. Sweeney [61] has proposed a general approach for doing k-anonymity for static database tables, aggregating data together into buckets to reduce identifiability. Another approach is to use mixes to make it harder to do traffic analysis (e.g. [9]). Confab currently does not have any built-in support for managing anonymity or for defeating traffic analysis, but could support these approaches in its architecture. Confab also provides support for application in which anonymity is not useful, for example, situations with family, friends, co-workers, and paid services.

In summary, while there have been many toolkits and infrastructures providing programming support and abstractions for sensors, and while there have been many individual techniques for managing privacy, Confab is the first to provide an extendable design that provides software architecture support and application developer support for building privacy-sensitive ubicomp applications that are optimistic, pessimistic, and mixed-initiative. Confab provides reusable mechanisms for both application developers and for end-users in managing personal information, as well as

mechanisms and abstractions for developers designing privacy-sensitive ubicomp systems.

6. FUTURE WORK

In the short-term, we plan on adding stronger support for proxies, to handle aggregation and k-anonymity as described in the previous work section. We are also planning on deploying a variety of sensors to capture a wide range of personal information and connecting them to Confab.

In the long-term, we plan on developing more applications and evaluating them with real users to assess how well people can understand the basic model of what the system knows about them and where their information is flowing, the privacy implications in sharing personal information, and overall quality and ease of interaction. As part of this ongoing work, we also plan on developing a set of GUI privacy widgets to make it easier to build the actual user interfaces of these applications. For example, the service description is a purely technical representation and is not one that end-users should interact with directly. We are looking into better representations and better interactions for common privacy needs.

7. CONCLUSIONS

We presented an extensive analysis of end-user needs and application developer needs for privacy-sensitive systems. The end-user needs were gathered through scenario-based interviews we did on location-enhanced applications, and on an analysis of surveys, research papers, message boards, proposed and existing privacy protection laws, and design guidelines for privacy-sensitive systems. The application developer needs were gathered through an analysis of research and commercial ubicomp applications.

These needs led to the high-level requirements of (1) a decentralized architecture, (2) a range of control and feedback mechanisms for building pessimistic, optimistic, and mixed-initiative applications, (3) plausible deniability built in, and (4) exceptions for emergencies.

We also presented Context Fabric, an infrastructure for building privacy-sensitive ubicomp applications for a spectrum of trust levels and privacy needs. From a software architecture perspective, Confab provides an extendable suite of operators that application developers and end-users can use for managing privacy. From an end-user perspective, Confab facilitates the creation of three basic interaction patterns for privacy-sensitive applications: optimistic, applications where the default is to share personal information and detect abuses; pessimistic, applications where it is more important to prevent abuses; and mixed-initiative, where decisions to share information are made interactively by end-users.

We described how Confab could be used to support the implementation of two privacy-sensitive applications.

8. ACKNOWLEDGMENTS

9. REFERENCES

1. AllNurses.com. http://allnurses.com/
2. Directive 95/46/EC. http://europa.eu.int/ISPO/legal/en/dataprot/directiv/directiv.html
3. Abowd, G.D., C.G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, Cyberguide: A Mobile Context-Aware Tour Guide. *Baltzer/ACM Wireless Networks* 1997. **3**(5): p. 421-433.
4. Adams, A. Multimedia Information Changes the Whole Privacy Ball Game. In Proceedings of *Computers, Freedom, and Privacy*. Toronto, Canada: ACM Press. pp. 25-32, 2000.
5. Addlesee, M., R. Curwen, S.H. Newman, P. Steggles, A. Ward, and A. Hopper, Implementing a Sentient Computing System. *IEEE Computer* 2001. **34**(8): p. 50-56.
6. AT&T, AT&T Wireless mMode - Find Friends. http://www.attwireless.com/mmode/features/findit/FindFriends/
7. Barkhuus, L. and A.K. Dey. Location-based services for mobile telephony: a study of users' privacy concerns. In Proceedings of *INTERACT 2003, 9th IFIP TC13 International Conference on Human-Computer Interaction*. To appear, 2003.
8. Bellotti, V. and A. Sellen. Design for Privacy in Ubiquitous Computing Environments. In Proceedings of *The Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*. Milan, Italy: Kluwer Academic Publishers, 1993.
9. Beresford, A. and F. Stajano, Location Privacy in Pervasive Computing, *IEEE Pervasive Computing*, vol. 2(1): pp. 46-55, 2003.
10. Brin, D., *The Transparent Society*. Reading, MA: Perseus Books, 1998.
11. Brown, P.J. and G.J.F. Jones, Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering. *Personal and Ubiquitous Computing* 2001. **5**(4): p. 253-263.

12. Burrell, J., G.K. Gay, K. Kubo, and N. Farina. Context-Aware Computing: A Test Case. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 1-15, 2002.
13. Cadiz, J. and A. Gupta, *Privacy Interfaces for Collaboration*. Technical Report MSR-TR-2001-82, Microsoft Research, Redmond, WA, 2001.
14. Castro, P. and R. Muntz, Managing Context for Smart Spaces. *IEEE Personal Communications* 2000. **5**(5).
15. Chen, G. and D. Kotz. Context Aggregation and Dissemination in Ubiquitous Computing Systems. In Proceedings of *Fourth IEEE Workshop on Mobile Computing Systems and Applications*. pp. 105-114, 2002.
16. Crowley, J.L., J. Coutaz, G. Rey, and P. Reignier. Perceptual Components for Context Aware Computing. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 117-134, 2002.
17. Cuellar, J., J. John B. Morris, D. Mulligan, J. Peterson, and J. Polk, Geopriv requirements (Internet Draft). 2003, IETF. http://www.ietf.org/internet-drafts/draft-ietf-geopriv-reqs-04.txt
18. Davies, N., S.P. Wade, A. Friday, and G.S. Blair. Limbo: A tuple space based platform for adaptive mobile applications. In Proceedings of *The International Conference on Open Distributed processing / Distributed Platforms (ICODP/ICDP '97)*. pp. 291-302, 1997.
19. Dey, A.K., D. Salber, and G.D. Abowd, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal* 2001. **16**(2-3): p. 97-166.
20. Doheny-Farina, S., The Last Link: Default = Offline, Or Why Ubicomp Scares Me, *Computer-mediated Communication*, vol. 1(6): pp. 18-20, 1994.
21. Edwards, J., Location Privacy Protection Act of 2001. http://www.techlawjournal.com/cong107/privacy/location/s1164is.asp
22. Edwards, W.K., M.W. Newman, J.Z. Sedivy, T.F. Smith, and S. Izadi. Challenge: Recombinant Computing and the Speakeasy Approach. In Proceedings of *Eighth ACM International Conference on Mobile Computing and Networking (MobiCom 2002)*, 2002.
23. Espinoza, F., P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund. GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In Proceedings of *Ubicomp 2001*. Atlanta, GA. pp. 2-17, 2001.
24. Falk, J., P. Ljungstrand, S. Björk, and R. Hansson. Pirates: Proximity-Triggered Interaction in a Multi-Player Game. In Proceedings of *Human Factors in Computing Systems: CHI 2001 (Extended Abstracts)*. pp. 119-120, 2001.
25. Federal Communications Commission, Enhanced 911. http://www.fcc.gov/911/enhanced/
26. Frelinghuysen, R., Wireless Privacy Protection Act of 2003. http://www.theorator.com/bills108/hr71.html
27. Garfinkel, S., *Database Nation: The Death of Privacy in the 21st Century*: O'Reilly & Associates, 2001.
28. Geocaching. http://www.geocaching.com/
29. Grimm, R., J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, *Programming for pervasive computing environments*. Technical Report UW-CSE-01-06-01, University of Washington Department of Computer Science and Engineering, Seattle, WA 2001.
30. Griswold, W.G., P. Shanahan, S.W. Brown, and R. Boyer, *ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing*. Technical Report CS2003-0765, Computer Science and Engineering, UC San Diego 2003.
31. Grudin, J., Desituating Action: Digital Representation of Context. *Human-Computer Interaction (HCI) Journal* 2001. **16**(2-4).
32. Gruteser, M. and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In Proceedings of *The First International Conference on Mobile Systems, Applications, and Services (MobiSys 2002)*, 2002.
33. Harper, R.H.R., *Why Do People Wear Active Badges?* Technical Report EPC-1993-120, Rank Xerox, Cambridge 1993.
34.
35. Hindus, D., S.D. Mainwaring, N. Leduc, A.E. Hagström, and O. Bayley, Casablanca: Designing Social Communication Devices for the Home. *CHI Letters (Human Factors in Computing Systems: CHI 2001)*, 2001. **3**(1): p. 325-332.
36. Hong, J.I., G. Boriello, J.A. Landay, D.W. McDonald, B.N. Schilit, and J.D. Tygar. Privacy and Security in the Location-enhanced World Wide Web. In *Workshop on Ubicomp Communities: Privacy as Boundary Negotiation (Ubicomp 2003)*. Seattle, WA, 2003. http://www.cs.berkeley.edu/~jasonh/publications/ubicomp2003-privacy-placelab.pdf
37. IBM Corporation, Enterprise Privacy Authorization Language (EPAL 1.1). http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/
38. Jiang, X., J.I. Hong, and J.A. Landay. Approximate Information Flows: Socially-based Modeling of Privacy in Ubiquitous Computing. In Proceedings of *Ubicomp 2002*. Göteborg, Sweden. pp. 176-193, 2002.
39. Johanson, B., A. Fox, and T. Winograd, The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing* 2002. **1**(2): p. 67-74.
40. Kaasinen, E., User Needs for Location-aware Mobile Services. *Personal and Ubiquitous Computing* 2003. **7**(1): p. 70-79.
41. Kindberg, T. and J. Barton, A Web-based Nomadic Computing System. *Computer Networks* 2001. **35**: p. 443-456.
42. Korba, L. and S. Kenny. Towards Meeting the Privacy Challenge: Adapting DRM. In Proceedings of *2002 ACM Workshop on Digital Rights Management*. Washington DC, USA, 2002.
43. Lamming, M. and M. Flynn. Forget-me-not: Intimate computing in support of human memory. In Proceedings

of *FRIEND 21: International Symposium on Next Generation Human Interfaces*. Meguro Gajoen, Japan. pp. 125-128, 1994.
44. Langheinrich, M. A Privacy Awareness System for Ubiquitous Computing Environments. In Proceedings of *Ubicomp*. Goteberg, Sweden. pp. 237-245, 2002.
45. Langheinrich, M. Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems. In Proceedings of *Ubicomp*. Atlanta, GA. pp. 273-291, 2001.
46. Lederer, S., J. Mankoff, and A.K. Dey. Who Wants to Know What When? Privacy Preference Determinants in Ubiquitous Computing. In Proceedings of *Extended Abstracts of CHI 2003, ACM Conference on Human Factors in Computing Systems*. Fort Lauderdale, FL. pp. 724-725, 2003.
47. Mayor, M., New Wireless Device Could Rescue Firefighters. 2001. http://www.wirelessnewsfactor.com/perl/story/9134.html
48. Nagel, K., C.D. Kidd, T. O'Connell, A. Dey, and G.D. Abowd. The Family Intercom: Developing a Context-Aware Audio Communication System. In Proceedings of *Ubicomp 2001*. Atlanta, GA. pp. 176-183, 2001.
49. Olsen, D.R., S. Jefferies, T. Nielsen, W. Moyes, and P. Frederickson, Cross-modal Interaction using XWeb. *CHI Letters, The 13th Annual ACM Symposium on User Interface Software and Technology: UIST 2000* 2000. **2**(2).
50. OnStar. http://www.onstar.com/
51. Palen, L. and P. Dourish, Unpacking "Privacy" for a Networked World. *CHI Letters (Human Factors in Computing Systems: CHI 2003)*, 2003. **5**(1): p. 129-136.
52. Pascoe, J. The Stick-e Note Architecture: Extending the Interface Beyond the User. In Proceedings of *International Conference on Intelligent User Interfaces*. pp. 261-264, 1997.
53. Povey, D. Optimistic Security: A New Access Control Paradigm. In Proceedings of *1999 New Security Paradigms Workshop*, 1999.
54. Priyantha, N.B., A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In Proceedings of *MobiCom 2000: The Sixth Annual International Conference on Mobile Computing and Networking*. Boston, Massachusetts: ACM Press. pp. 32-43, 2000.
55. Rhodes, B. and T. Starner. The Remembrance Agent: A Continuously Running Automated Information Retrieval System. In Proceedings of *The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96)*. London, UK. pp. 487-495, 1996.
56. Román, M., C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing* 2002. **1**(4): p. 74-83.
57. Schilit, B.N., *A Context-Aware System Architecture for Mobile Distributed Computing*, Unpublished PhD, Columbia University, 1995. http://seattleweb.intel-research.net/people/schilit/schilit-thesis.pdf
58. Schilit, B.N., N.I. Adams, and R. Want. Context-Aware Computing Applications. In Proceedings of *Workshop on Mobile Computing Systems and Applications*. Santa Cruz, CA: IEEE Computer Society, December 1994, 1994.
59. Sloane, L., Orwellian Dream Come True: A Badge That Pinpoints You, *New York Times* p. 14, 1992.
60. Spreitzer, M. and M. Theimer. Providing location information in a ubiquitous computing environment. In Proceedings of *Fourteenth ACM Symposium on Operating System Principles*. Asheville, NC: ACM Press, December, 1993.
61. Sweeney, L., k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 2002. **10**(5): p. 557-570.
62. Talbott, S., The Trouble with Ubiquitous Technology Pushers, or: Why We'd Be Better Off without the MIT Media Lab. 2000. http://www.oreilly.com/people/staff/stevet/netfuture/2000/Jan0600_100.html
63. Want, R., A. Hopper, V. Falcão, and J. Gibbons, The Active Badge Location System. *ACM Transactions on Information Systems* 1992. **10**(1): p. 91-102.
64. Weiser, M., R. Gold, and J.S. Brown, The Origins of Ubiquitous Computing Research at PARC in the Late 1980s. *IBM Systems Journal* 1999. **38**(4): p. 693-696.
65. Westin, A.F., *Privacy and Freedom*. New York NY: Atheneum, 1967.
66. Whalen, J., You're Not Paranoid: They Really Are Watching You, *Wired Magazine*, 3(3): pp. 95-85, 1995.