User Modeling and User-Adapted Interaction (2005) 15:197–234
DOI 10.1007/s11257-005-0001-z

Consistent Modelling of Users, Devices and Sensors in a Ubiquitous Computing Environment

DAVID J. CARMICHAEL, JUDY KAY and BOB KUMMERFELD
School of Information Technologies, University of Sydney, Australia 2006.
e-mail: {dcarmich,judy,bob}@it.usyd.edu.au

(Received: 26 April 2004; accepted in revised form 7 February 2005)

Abstract. This paper describes the use of an `accretion-resolution` user modelling representation to model people, places and objects. We explain the motivation for the key properties of the representation, especially those of particular importance for ubiquitous computing: firstly, for flexibility in interpreting the typically noisy and potentially conflicting evidence about users' locations; secondly, to support users in scrutinising their user model, the processes that determine its contents and the way that it is used in the ubiquitous computing environment.
A novel and important aspect of this work is our extension of the representation beyond modelling just users, using it also to represent the other elements such as devices, sensors, rooms and buildings. We illustrate our approach in terms of models we have been building for a system which enables users to gain personalised information about the sensors and services in a ubiquitous computing environment. We report experiments on the scalability and the management of inconsistency in modelling of location, based on `accretion-resolution`.

Key words. modelling location, modelling pervasive computing environments, scrutability, user control, user model representation

1. Introduction

User modelling has an important role in ubiquitous computing. It is essential for important forms of the personalisation of user environments: it is user models which will be the repositories of information that might be collected about a user from ubiquitous sensors. Modelling the user's location is central to much ubiquitous computing work.

This is reflected in the large amount of work on mechanisms that can be used to determine a user's location. Many of these operate invisibly, from the early active badges (Want et al., 1992) to the now common radio-based sensors like Wi-Fi and Bluetooth, low cost radio-frequency tags and associated RFID readers as well as ultrasound devices as in the Cambridge BAT (Addlesee et al., 2001) and in the Cricket system (Priyantha et al., 2000). In addition, many other sensors, such as cameras, pressure pads and microphones may sense people. To model a person's location, a system must interpret the data from such sensors to model aspects of the user that are associated with location.

We want to model the user's location, activity and other relevant aspects of context in a manner that is consistent with the other information a system holds about a user in a user model. We introduce our approach in terms of two, closely related, target classes of scenario: the first is a personalised user interface that enables each user to *see* all elements of the ubiquitous computing environment, especially the *invisible* ones; and the second is to support one person in locating another person.

1.1. THE INVISIBILITY SCENARIO

> Alice walks into her house, and pauses at the mirror in the foyer, checking her hair. Then she asks for any messages and it lights up, displaying messages that her children have left for her. She walks into the lounge room and the new release of Monsieur Camembert begins playing. Later, her father enters the house. He is visiting for a few days and staying at the house. For him, the mirror in the foyer behaves only like a conventional mirror. He is rather irritated at the music coming from the lounge, but is then surprised when it changes to a recording of his grandchildren's recent band concert. (He does not like this music but has always tried to indicate that he does.)

This scenario illustrates the *invisibility* goal of ubiquitous computing environments, where devices should fit so well and operate so naturally, that the experienced user, like Alice, finds that they blend into the environment. Another goal of invisibility is that the computing elements in the environment should be *unobtrusive*. This is the case with the sensors that are needed to detect Alice for the mirror and music in the scenario to operate.

This naturalness and unobtrusiveness can pose problems. It may be impossible for people even to be aware of facilities available to them. Once Alice knows about the mirror and how to activate it, it may provide a natural and convenient hands-free interface. However, for the new user, even locating the interface may pose real problems. We can expect this to be a long term issue of importance because invisibility is one of the fundamental goals of pervasive computing design.

There are also problems in unobtrusiveness of *sensors*, such as those used in well-known ubiquitous computing projects, like the Georgia Tech Aware Home (Kidd et al., 1999) and the Microsoft Easy Living Project (Brummit et al., 2000), as well as many others. This is in conflict with privacy principles such as those of Langheinrich (2001), which require that people be able to determine what sensors operate in an environment and what they do with the information they collect. It also is at odds with the observation (Ackerman et al., 1999) that many people want to control the use of personal data, especially where that data can be linked to their identity. This makes it important that a user model representation in this environment supports such scrutiny.

Another important aspect of the scenario relates to the personalisation of services and facilities available. In the scenario, the mirror's hidden services were not available to Alice's father. The different choices of music delivered are an example

of personalising the environment, first to Alice's preferences when she was alone, and then, when her father arrived, it switched to music which the system predicted both she and her father would enjoy. Since her father was surprised at the change in music, and did not like it, the scenario illustrates a case where a person may wish to be able to determine how the environment chose the music and how he might correct this in the future.

This scenario illustrates several issues, and in this paper we focus on the need to provide support for the user to determine:

- what facilities are available and relevant to them in their current environment;
- what sensors operate in an environment;

This involves the typical long term elements of user modelling, where the system maintains a model of aspects of the user. It also calls for maintenance of the user's dynamic location. We now consider another scenario that requires location modelling to help users interact with each other.

1.2. LOCATOR SCENARIO

> Boris is an academic who carries a Bluetooth enabled PDA. Natasha is a student who comes to his office, wanting to discuss a problem about her assignment in Boris's course. She consults the `Locator` interface at the door to determine where he is, recording a voice message explaining the need to meet him. `Locator` asks her to wait while it locates Boris.

We now consider three possible cases of what happens next:

1. Boris has set `Locator` to indicate that he is willing to be contacted by people who come to his door. The `Locator` system consults the user model for Boris and is able to detect that he is talking with researchers in the lab nearby. `Locator` calls the phone in that lab, delivering the audio message from Natasha. He indicates that he is coming to meet her. The `Locator` system tells Natasha that Boris is coming and she should wait. Within a few minutes, Boris comes to meet her.
2. The system determines that its best information about Boris's location is that 20 minutes ago he was at his home, many kilometres away. It informs Natasha that he is not available.
3. Boris is in his office, working at his computer. He has established a `Locator` policy that he is not to be interrupted when he is actively using his computer, meaning that he has used the keyboard or mouse within the previous five minutes. The `Locator` system tells Natasha that Boris is not available.

Note that users think of locations in terms of places or rooms that make sense in this particular social context, rather than more rigid but precise representations like longitude and latitude. This set of scenarios illustrates the following issues.

- Depending upon the situation, an application may need different levels of 'spatial accuracy' in the modelling of Boris's location. For example, in the first case, the application needed to determine the room he was in. By contrast, in the second, it does not: it needs only to model that he is not near. In neither case was it necessary to model location to the fine detail such as in active badge systems (Addlesee et al., 2001; Scott and Hazas, 2003) and ultrasound-based systems like Cricket (Priyantha et al., 2000).
- Similarly, different applications require different 'time accuracy'. For example, since `Locator` determines that Boris was at home 20 minutes ago and that it typically takes him at least 30 minutes to travel that route, then it can declare him unavailable. On the other hand, it needs much finer resolution, of the order of a few minutes, on locations within the building.
- User location may depend upon who asks for the information. In the third case, Boris was unavailable. However, if Boris's daughter was at the door, rather than Natasha, `Locator` might tell her to wait (and it would interrupt him).
- Location determination must be fast enough for Natasha to wait for an indication that Boris is either unavailable or that it is trying to contact him.

We now describe our approach to defining, implementing and evaluating a user modelling representation that supports the creation of ubiquitous computing environments that address these issues. In Section 2, we give an overview of our representational approach in the case of modelling a user's location and in Section 3, we describe the `MyPlace` prototype for location modelling in support of scenarios like the ones just described. We then return to the two scenarios: in Section 4 we give examples of the information provided to the user in relation to a simpler form of the invisibility scenario; and in Section 5 we report experiments showing how our user model representation supports the needs of the `Locator` scenario. Section 6 reports experiments on scalability and the management of inconsistency. Section 7 describes related work on user model representations, linking it to the challenges of ubiquitous computing environments and the modelling of location and Section 8 has the final discussions, conclusions and projections of future work.

2. Overview of the Accretion-Resolution Representation for Modelling user Location

We introduce the `accretion-resolution` representation for user modelling for systems which implement the scenarios described above. The representation has two basic operations. The first, *accretion*, involves collection of uninterpreted evidence about the user. Each piece of evidence includes the *source* and the *time* associated with it. The second operation is *resolution*, the interpretation of the current collection of evidence when we need to know the user's location or the values of other aspects of the user model.

We now illustrate the `accretion-resolution` approach used to model location. The description is in terms of a single user, Bob. Figure 1 shows part of Bob's work environment with its four Bluetooth sensors and one activity sensor. The Bluetooth sensors can detect Bob's mobile phone, when it is within the sensor's range.

Table I illustrates the type of data that is collected in this environment. Each column corresponds to data coming from a sensor which can contribute evidence about Bob's location. It is based upon actual data we have collected from a variety of sensors. A subset of that data is shown in the Appendix A. Each asterisk (*) in Table I indicates that the sensor in that column sent a piece of location evidence at the time shown for that row. Since there are several hundred pieces of evidence about Bob's location on that day, we show a small selection of those that are useful for describing the `accretion-resolution` approach.

Figure 1 shows a Bluetooth sensor is near the main entrance foyer. This often detects Bob's Bluetooth enabled phone as he arrives at work and departs at the end of the day. Also, this is a quite popular place for people to meet and chat and it is en route to various places Bob needs to go during the work day. So Bob may be detected by it at various times during the day. Since there are many entrances to the building, this sensor will not detect him when he uses

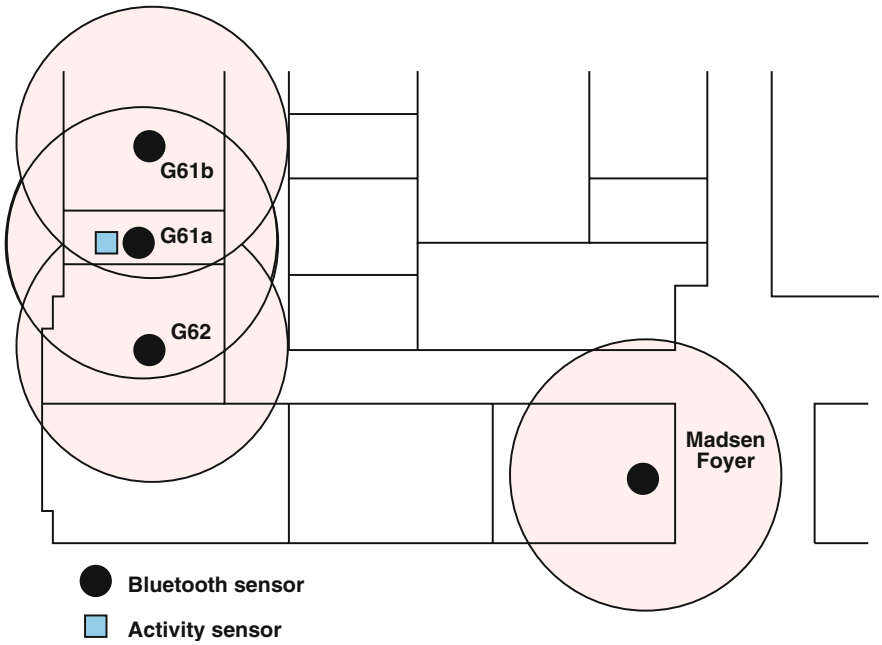

Figure 1. Map of sensors in lab area for current prototype. Bluetooth sensors are shown as filled circles and their approximate range is shown. Not shown here is the Bluetooth sensor that is in the Seminar room which is in another part of the building. Also, not shown are the sensors in a home, several kilometres from the building. The shaded square is an activity sensor.

Table I. Example of the selected evidence available about Bob's location

Time		Home		Work						
ID	Actual	Office A	Lounge B	Foyer B	Office B	Office A	LabA B	LabB B	Sem B	Location
01	06:54:38		*							Home
02	06:58:46		*							
03	09:40:13			*						Foyer
04	09:41:55							*		Office
05	09:42:02				*					area
06	09:42:04						*			
07	09:44:17					*				
09	09:49:12				*					Office
10	09:49:13						*			area
11	09:50:06							*		
12	12:10:04					*			*	Seminar
13	13:16:27								*	room
14	20:10:24	*								

Each column corresponds to a different evidence source. *A* indicates machine activity sensors, *B* indicates Bluetooth-based sensors.

one of those other entrances. Also, this Bluetooth sensor is prone to miss detecting Bob. We conjecture that this may happen if there are many people in the foyer. This type of problem is typical of Bluetooth sensors for detecting mobile devices, like phones: they have limited range, in this case about 10 metres; they can be slow to detect Bluetooth devices and may fail to detect them, causing false negatives.

Figure 1 also shows the three Bluetooth sensors near Bob's office (g61a), one actually in it and one in each of the adjoining laboratories (g61b and g62). The range of our Bluetooth sensors is about ten metres, and the Bluetooth sensor in Bob's office is roughly ten metres from the other two. Since there are walls between the sensors and there are varying numbers of people within each room, it turns out that when Bob is actually in his office, he will be reliably detected by the Bluetooth sensor in his office and often detected by one or both of the other two. When Bob is in either of the labs, he is reliably detected by the relevant lab Bluetooth sensor and sometimes by his office sensor. This situation is also typical of this type of sensor and needs to be taken into account when reasoning about Bob's location.

Another important factor is that the Bluetooth sensors are sometimes moved, perhaps over periods of several weeks. It is important to be able to easily model such changes. In addition, if a system is to support historical location questions, such as determining Bob's location a month ago, it must take account of such changes in the location of sensors.

In addition to the Bluetooth sensors, there is an activity sensor in Bob's office (see Figure 1) and at his home. These detect mouse and keyboard activity on his computer. The corresponding columns of Table I are labelled A.

The last column of Table I shows one interpretation of Bob's location, based on the available data from the sensors. The first piece of sensor evidence comes from the Bluetooth sensor in Bob's lounge. We see that at times 1 and 2, Bob is detected in his lounge.

The table shows six sources of data about Bob's location at work. The Bluetooth sensor in the main foyer shows him arriving at work at time 3. In the time periods 4 – 7, Bob is detected by the sensors in his office or in the two labs near his office. Given that these times are so close, we can be reasonably confident Bob is in the area of his office. Moreover, if we are asked to account for this conclusion, it could be explained on the basis of the evidence from the sensors in his office and the nearby labs.

Note that if we need to know a more precise location, the interpretation is more complex. For example in time period 7, Bob appeared to be active at his machine and was detected by his office Bluetooth sensor about 2 minutes earlier. He was also detected by each of the two nearby lab sensors in the previous 2–3 minutes. The interpretation would be easier if we had additional knowledge about the likelihood that these sensors would detect Bob when he was actually in his office.

Table I also illustrates another interesting case. In time period 12, Bob's office activity sensor is activated at exactly the same time as the seminar room Bluetooth detector senses him. Since the seminar room is more than 50 metres from his office, it is impossible for him to be in his office and detected by the seminar room sensor. Even if we take account of the possibility that the different clocks in different parts of our system have some small inaccuracies, this pair of pieces of evidence appear to be in conflict.

There are many ways to resolve this conflict. Depending upon the nature of the location modelling required, we might use various strategies to conclude a value. We discuss this below, in Section 3. For example, Bob might know that his machine is sometimes used by other people in the lab. There are many other possible explanations for the apparent inconsistency in location model. For example, he may often lend his phone to other people. This type of problem is inherent in location modelling based upon sensing such devices.

The example illustrates the approach taken to location modelling using the `accretion-resolution` representation, as implemented in the `Personis` user modelling system (Kay et al., 2002). There are four main elements:

- *sensors*, detecting the user;
- *transformation* of the raw sensor data into evidence for use in reasoning about location;
- *accretion* of such evidence, ignoring conflicting information at this stage;

- *resolution*, a process that interprets evidence to determine the user's location at the time that an application requires it.

2.1. IMPLEMENTATION OF THE ACCRETION-RESOLUTION REPRESENTATION

We have built three main implementations of the `accretion-resolution` representation. Our experience with the earlier ones has informed the current work. First, we built the `um` toolkit (Kay, 1995) and used it extensively to model users of a text editor. The second main implementation of the `accretion-resolution` representation is the Personis user model server (Kay et al., 2002). This version provided flexible support for defining *personas*, with information about users controllable at several levels: view, component, evidence from nominated sources and the `resolvers`. The third implementation, used in this work, is a library within an application. It is intended for use in mobile devices where memory and processing constraints are tight.

3. Overview of the MyPlace Prototype

To evaluate our approach, we have implemented a prototype ubiquitous computing environment which we call `MyPlace`. This is a testbed for personalised ubiquitous computing environments that support scrutability of the personalisation. We now describe the core elements of the approach: the accretion of location evidence in device and sensor models; location modelling based upon the evidence; the conflict resolution process used to deal with conflicting evidence and to provide flexibility in temporal, spatial and privacy granularity; as well as details of the implementation.

3.1. ACCRETION OF LOCATION EVIDENCE IN DEVICE AND SENSOR MODELS

The part of the architecture that is responsible for building device and sensor models of `MyPlace` is shown in Figure 2. At the top, it shows the range of sensors. At the bottom are the `Personis` models, in this case:

- sensor models for entities such as the foyer Bluetooth sensor.
- device models for entities such as Bob's Bluetooth enabled phone;

The middle of the figure shows the Publish/Subscribe server. `MyPlace` establishes a subscription for all messages related to sensors and devices. Then, it accepts incoming data from the Publish/Subscribe server and uses that to update the device and sensor models, as we now describe.

Bluetooth sensors perform a scan every 30 seconds, detecting Bluetooth devices in their environment. Whenever a sensor detects a device, it publishes a message to the *Publish/Subscribe server*. This message includes the detected device's Bluetooth ID and the sensor's own ID. Every sensor and device has its own, unique ID. For

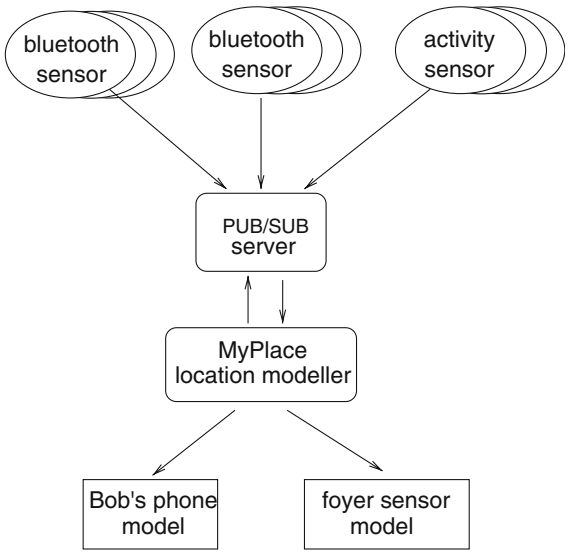

Figure 2. Accretion elements of the `MyPlace` architecture.

example, if the foyer sensor detects Bob's phone, the message has the unique ID for the foyer sensor and Bob's phone.

An activity sensor operates in a similar manner. It scans for mouse and keyboard activity in the last 5 minutes. If there is no activity, it does nothing but if there is activity, it publishes a message to the *Publish/Subscribe server*. This message has the sensor ID and the ID of the owner of the sensor. For example, the owner of the activity sensor on Bob's computer is *Bob*. As in the case of the Bluetooth sensors, each sensor and device has its own, unique ID. In general, all sensor messages, from any form of sensor, contain:

- *SourceID*: ID of the sensor.
- *Type*: which is always *sensor*.
- *SenseeID*: ID of the thing sensed.
- *SensorType*: *Bluetooth*, *system activity*, etc.
- A timestamp.

Figure 3 shows how sensor data contributes evidence to the sensor and device models when Bob's Bluetooth phone is detected in the foyer.

Step A occurs when the `MyPlace` *location modeller* first starts up. It subscribes with the *Publish/Subscribe server* for all messages of type *sensor*. The `MyPlace` *location modeller* has to check for network failures and re-establish this subscription as necessary.

Step B shows how the Bluetooth sensor in the foyer acts upon detecting Bob's phone. Importantly, it does not recognise that the phone belongs to Bob. The

A. `MyPlace` *location modeller* subscribes for messages of *Type* = *sensor*.

B. *Bluetooth foyer sensor* detects <*BobsPhone MAC address*>. It sends (publishes) a message:

SourceID : <*ID for sensor in foyer*>
Type : *sensor*
SenseeID : <*BobsPhone MAC address*>
SensorType : *Bluetooth*

C. Publish/Subscribe server receives this message and sends it to `MyPlace` *location modeller*.

D. `MyPlace` *location modeller*, creates two pieces of evidence, each with the current time stamp:

•*foyer* sensor model - *accretes* evidence that sensor has *Seen* <*BobsPhone MAC address*>
•*BobsPhone* device model - *accretes* evidence that device was *Seenby* <*foyer sensor*>.

Figure 3. Process for sending sensor data into models for devices and sensors, where a device with ID <*BobsPhone MAC address*> is detected by the *foyer* sensor. Angled brackets, as in <*BobsPhone MAC address*>, indicate this is a description rather than the actual value.

message simply records the device's unique ID (*MAC address for the phone*) as the *SenseeID*, and the other elements shown in the Figure.

In Step C, the *Publish/Subscribe server* receives this message. From Step A, it has a subscription from the `MyPlace` *location modeller* for this message. So, it sends the message on to the `MyPlace` *location modeller*.

Step D is the *accretion* stage that occurs when the `MyPlace` *location modeller*, uses the message to update both the sensor and device models:

- the model for the *foyer* sensor accretes one piece of evidence that it has sensed a device with ID ⟨*BobsPhone MAC address*⟩. We call this model component *Seen* since it models devices this sensor has seen.
- a corresponding piece of evidence is stored in the device model's *Seenby* component, which models what sensors this device has been seen by.

Essentially, this means that the sensor model remembers what it has seen and the device model remembers that it has been seen by the sensor.

3.2. LOCATION MODELLING

The *accretion* process described above is continuous, with evidence added as each sensor detects a device. We now describe what happens when an application asks for a person's location. Figure 4, shows how an application, such as `MyPlace`, reasons from sensor and device models to determine Bob's location in his user model. The figure has numbered lines corresponding to steps needed to determine Bob's location, in the case where Bob's phone was sensed by the foyer sensor.

At the top left of the figure, is the user model for Bob. The figure shows just a few of the relevant components of his user model:

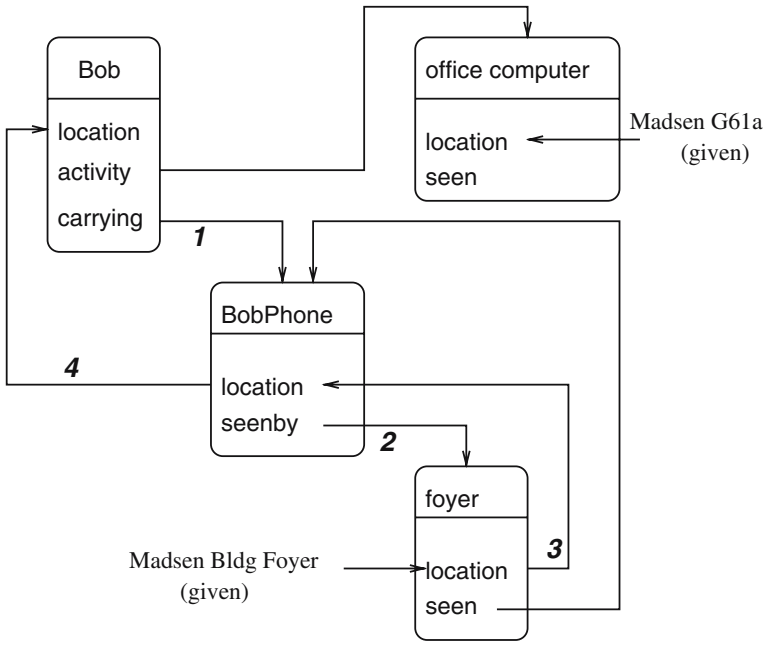

Figure 4. Example of the way that the evidence from the foyer sensor is used to contribute to the model for Bob's location. The query for Bob's location uses the *carrying* component of his model to determine the devices he is carrying, in this case his phone. It then consults the phone device model to determine what sensors this was *Seenby*, in this case, the foyer sensor. It then consults the foyer sensor model to determine its location which adds evidence on the location of Bob's phone. Finally, the evidence is added to Bob's model.

- *location* modelling his location;
- *carrying* representing devices Bob is carrying;
- *activity* for his activity sensors (such as the one on his office computer).

When `MyPlace` wants to know Bob's location, it first *ask*s his user model for the list of devices he is *carrying*. In the figure, we indicate that there is evidence, which is interpreted by Personis, to mean that one of these is Bob's phone.

So, `MyPlace` queries the device model for Bob's phone. This is the line labelled **1** in Figure 4, to indicate that the value in Bob's user model is used to direct a query on the device model for his phone. That device model has a component *location* for its own location. It also has a *seenby* component. This accretes evidence from sensors (via the Publish/Subscribe server) as described in the last section. In our example, we suppose that the query on Bob's phone device model, for the last location where this device was detected, returns the ID for the foyer sensor.

At this stage, `MyPlace` queries the model for the foyer sensor. The figure illustrates this with the line labelled **2**.

The model for the sensor, shown in the figure as *foyer*, has a component for its *location* which is the foyer. So when `MyPlace` queries its location, this value is returned and `MyPlace` lodges a piece of evidence in the *location* component of

the phone model, to indicate that its location at that time was in the foyer. This action is shown by the line labelled **3**. This means that the phone model keeps a trail of all such queries on the phone's location.

Finally, `MyPlace` uses this value to add a piece of evidence to Bob's user model. This is evidence from the sensing of phone's location at the time that phone was last seen in that location. This step is indicated by the line labelled **4**.

The figure shows that Bob's activity sensor is also linked to his model. To keep the figure simple, we have not shown the flow of reasoning and evidence for it. We do note, however, that the activity sensor is directly associated with Bob, since he does not carry it.

3.3. RESOLVING THE VALUE OF A PERSON'S LOCATION

The process described above may produce many pieces of evidence, one for each of the devices that the user model indicates that the user is carrying as well as for non-mobile sensors such as the activity sensor. These may conflict, with some evidence indicating Bob is in one place and other evidence that he in another place. The last stage in determining Bob's location is to reconcile this set of evidence, resolving any inconsistencies.

The `accretion-resolution` representation allows for arbitrary *resolver*s. These may use a simple Point algorithm (Hightower and Borriello, 2004). This estimates location based on the most recent piece of evidence. Hightower reports that it is 'used by most commercial infrared badge systems as well as some cellular telephony location services'. Equally, a resolver can implement a sophisticated algorithm such as a particle filter (Hightower and Borriello, 2004) or Dynamic Bayesian Networks (Patterson et al., 2004). Of course, there are also many other algorithms of intermediate sophistication; Hightower describes the Centroid, which combines a small set of the recent data, the Smooth Centroid which is similar, but weights more recent evidence more heavily and the Smooth Weighted Centroid which also takes account of device characteristics. A similar approach could be based upon the `accretion-resolution` evidence classes (Kay, 1995) combined with time.

Importantly, the *resolver* is applied at the time the application requests the user's location: so, at this stage, less privileged applications, such as in the case of the `Locator` scenario, can only use restricted *resolvers*.

3.4. SEPARATION OF MODELLING EVIDENCE ACROSS SENSOR, DEVICE AND USER MODELS.

We need to explain the motivation for keeping separate sensor, device and user models. Certainly, an alternative architecture could simply place all evidence directly into the user model. So long as that evidence was timestamped and indicated the sensors that provided it and the device detected, the reasoning

process could be simplified by avoiding the need for the four steps shown in Figure 4. There are several reasons that our architecture does not do this.

One important reason is that our approach affords distribution of the data and computations. This is important for scalability and efficiency.

Another reason is to handle movement of sensors. When a sensor is moved, a new piece of evidence is given about its location. This component of the model effectively maintains a history of the location of this sensor. This makes it possible to perform queries about people's locations in the past: in that case, the relevant time is matched against the sensor's location at that time. This representation associates a unique ID with each sensor and then uses evidence about that sensor's location to map the ID to location. This mapping of sensor to location correctly handles the case where sensor locations change. This will be particularly common when sensors are also mobile.

An important reason for maintaining an explicit sensor model is that it keeps all information about that sensor in one place. For example, although Figure 4 omits such details, the device models can model who is carrying them, with a component that corresponds to the *carrying* component in user models. It is then straightforward to answer such questions as: who was the last person to leave via this door?

Similarly, maintaining a separate device model supports queries such as: what sensors have recently detected this device? Answering this in our example requires the *seenby* component of the phone model and the location of the sensor, as shown in Figure 4. This is important for the Invisibility Scenario because we want to be able to present information about the environment using devices that the user is carrying.

Note that the evidence in Bob's *location* component is used for more than simply calculating his location. Recall that this evidence is only added to his location model when there is a request for his location. This piece of evidence includes details of the application that requested his location. We designed the representation so that his user model effectively has an audit trial of all requests for his location. If Bob wants to scrutinise the way that his model has been used, he can examine this information. Importantly for scrutability, we ensure that the user model is not full of large numbers of sensor detections that were never used by any application. At the same time, historical queries can always be made, following the process shown in Figure 4, but requesting resolved values for the time of concern, rather than simply the most recent evidence as in our example.

3.5. IMPLEMENTATION OF MYPLACE

The software running in the processors controlling the Bluetooth sensors is written in C. Currently, the sensors are located in two buildings, the workplace and a home as in the example above. The activity sensors run on two personal machines, and as in the example, one of these is in the workplace and one in a home. Sensors send their messages to the *Publish/Subscribe server*, which is implemented

with Elvin (Segall et al., 2000). This runs on a machine in the workplace building being modelled. The location modeller programs can run on any machine, having established a subscription at the *Publish/Subscribe server*. The particular location modeller program reported in the experiments below is implemented in Python and runs on a separate machine in that workplace building. The user, sensor and device models are stored on another machine (for pragmatic reasons). In the current implementation messages from sensors are not encrypted. Personis uses a backend database, Berkeley DB[1] for the user model evidence.

4. MyPlace Interface for the Invisibility Scenario

We now describe the interface which makes use of the `accretion-resolution` representation and the `MyPlace` architecture to enable users to determine the sensors and facilities in the environment, the core of the invisibility scenario. The `MyPlace` invisibility interface presents information about a building, based upon the identity of the user and their location. Figure 5 shows examples of the interface seen by two users when they first enter the building. Fred is a new graduate student. John is a visitor to the building and he is coming to present a seminar.

The difference between what is delivered to the two users depends upon which sensors and devices are relevant to each. The system starts by telling the user where it thinks they are. It then lists the ID for the one sensor (where this is its MAC address *00:80:C8:35:53:6B*) in the current location and provides a link to a list of other sensors in the building. To this point, the displays are the same for both users.

Figure 5. Two user's view of the whole building.

[1] http://www.sleepycat.com/

The difference between the two displays is in the details of other locations that are nearby. John is only shown information about the seminar room (g92). By contrast, Fred sees information about his workspace (g61b), the departmental resource room (G70) and the seminar room (g92).

In Figure 6, we contrast views of a room as presented to Fred and Bob, an academic who is Fred's supervisor. In this case, notice that Fred is not allowed to use the colour printer and thus is not shown it. Bob's location listing also includes his own office in addition to the places presented to Fred.

Our building and room models are implemented using `Personis` and so are represented in a manner that is consistent with the other elements of the system.

Note that this approach can also be applied where users do not want to disclose their identity. In that case, the user is informed about all Bluetooth sensors that can detect them as well as any public services in the building. There are also many other possible personas that can be supported. For example, since this building is used by undergraduate computing students as well as geography students, a user could take the *persona* of a geography undergraduate and see the services available to such students. Equally, the same student could decide to take on a computing undergraduate *persona* and be presented with details of services available to them. (We note that in the scope of this paper, we cannot deal with the associated security issues and the need to authenticate identity, although we acknowledge that these are important issues in the practical deployment of systems.)

Location or world models are a crucial part of many context aware applications. There has been a range of work on such models, including, for example, Project Aura's Context Information Service (Judd and Steenkiste, 2003) and the Nexus

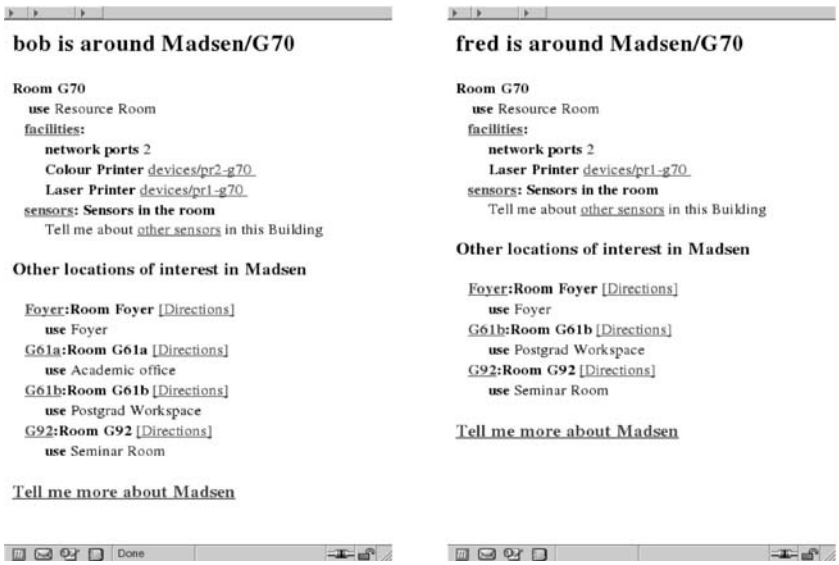

Figure 6. Two user's views of a particular room.

system (Bauer et al., 2002) which has been integrated in the Georgia Tech Aware Home (Lehmann et al., 2004).

The distinctive aspect of our work is the generalised use of our user modelling representation for rooms and spaces so that these are managed consistently, extending the scrutability of the user model to the other entities in the pervasive computing environment.

In our introduction to the invisibility scenario, we noted the need for personalised descriptions of the user's environments. The figures in this section illustrate how `MyPlace` enables users to see what is in their current location, so that they can *see* the otherwise invisible *sensors* that are detecting them as well as the *facilities* available to them.

5. Fine-Grained Single User Evaluation: The Locator Scenario

We now describe the use of the representation to manage the issues identified in the `Locator` scenario:

- flexible *spatial accuracy* location modelling;
- flexible *time accuracy* location modelling;
- selective presentation of location depending upon *who asks*;
- *timely* response to location queries

In addition, we need to demonstrate how `accretion-resolution` deals with the issues, identified in Section 2, in modelling sensors:

- sensors may be moved;
- they have varying accuracy in terms of range, error rates, likelihood of false positives and negatives;
- different sensors may provide inconsistent location information and this must be interpreted, taking account of the fact that different user behaviours require different policies in this interpretation.

We now present results of our experiments in relation to these issues. Our results are based upon sensor data for one user, Bob, and the devices and sensors needed to model his location:

- one carried device, a Bluetooth enabled mobile phone,
- two activity sensors, one at his home and one in his work office
- and six Bluetooth sensors, one at his house and the remainder in his workplace. The sensors correspond to those of Figure 1, Table I and the Appendix A.

The user whose results are reported here is one of the authors of the paper. Currently fifteen people are registered with `MyPlace`. None has as extensive a collection of sensor detections as the user whose results are reported here.

Figure 7 shows the Bluetooth evidence for the user's location through one typical work day. This corresponds to making only evidence that came from Bluetooth evidence sources visible to the `resolver`. This graph shows each evidence point connected with a line. So, for example, there is evidence from the sensor at Bob's home at around 7 am. Then there is a period where no evidence is available until around 9.40 am when he is detected by the sensor in the foyer of his workplace. There are considerable periods when he is in the area of his office. This appears as evidence from the sensors in his office (g61a) and the two adjoining labs (g62 and g61b). There are two blocks of time when he is detected by sensors in the seminar room (g92). Since the lines in the graph connect evidence points, the horizontal lines may equally well reflect a series of many evidence points, all from the same location, or just two points of evidence, one each at the beginning and end of the lines.

Figure 8 shows Bob's location on the same day, as assessed by a two-value `resolver`: this returns only *work* for all evidence from sensors at work and *unknown* where there is evidence from any other source. Similar to Figure 7, this graph has one point for each resolved location value and lines connecting these. So, for example, we see that as soon as Bob is detected by a work sensor, his resolved location is shown as work until there is evidence for his location elsewhere. Both Figures 7 and 8 show one point for each piece of evidence in any of the relevant sensor models. In practice, we are more interested in the location requests from applications and these will require a resolved location value based upon the sensor evidence available over a specified recent period of time. We illustrate this in the next set of graphs.

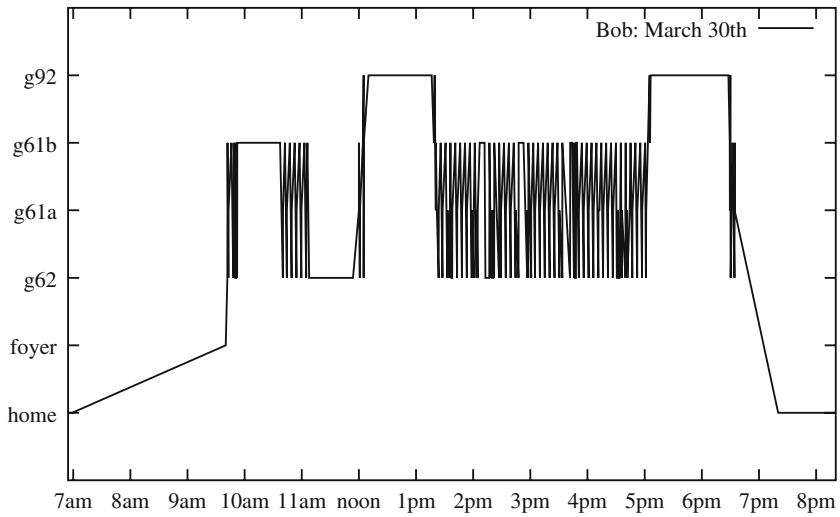

Figure 7. Bluetooth evidence for one user, Bob, on March 30th. The modelled locations are on the *Y*-axis and time on the *X*-axis. Each piece of evidence is a point in the graph. Lines connect these. There was no sensor data for the user before the first point at appropriately 7 am or after the last point at approximately 7.30 pm.

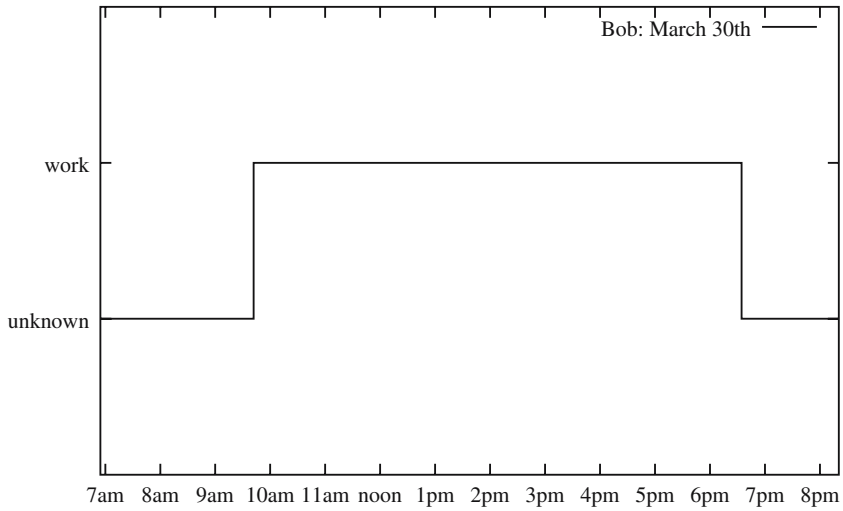

Figure 8. Bob's location based on the same evidence as in Figure 7 using a `resolver` which only indicates whether he is at work or not.

Figure 9 shows how we model whether Bob is *active*, based on the policy described in the third case of the `Locator` scenario. This policy defines the user (Bob in this case) as unavailable if he has used the keyboard or mouse of his work terminal in the last five minutes. The reason for making this time 5 minutes is that the activity sensor only detects mouse and keyboard activity, and the policy is intended to allow for the periods, of up to 5 minutes, when he is focussed on a task but is in-between active use of the mouse or keyboard. The figure shows that this policy establishes periods when Bob is actively working and the `Locator`-style interface would not disturb him at these times. Of course, by this policy, if Bob is not detected as active in the last 5 minutes, we can only conclude that we do not know whether he is active.

Figure 10 shows the same type of Bluetooth evidence as in Figure 7 but it covers a six day period. (There was no sensor data on the seventh day.) Figure 11 is for the same period as Figure 10 but it shows his availability to a general user who comes to his door wanting to talk with him. The figure was generated by querying the user model for Bob's availability every minute, according to the following policy:

available
    if a sensor has detected him in the area near his office
        (g61, g61b, g62) in the last 10 minutes and
    he has not been active in the last 5 minutes;
unavailable
    otherwise.

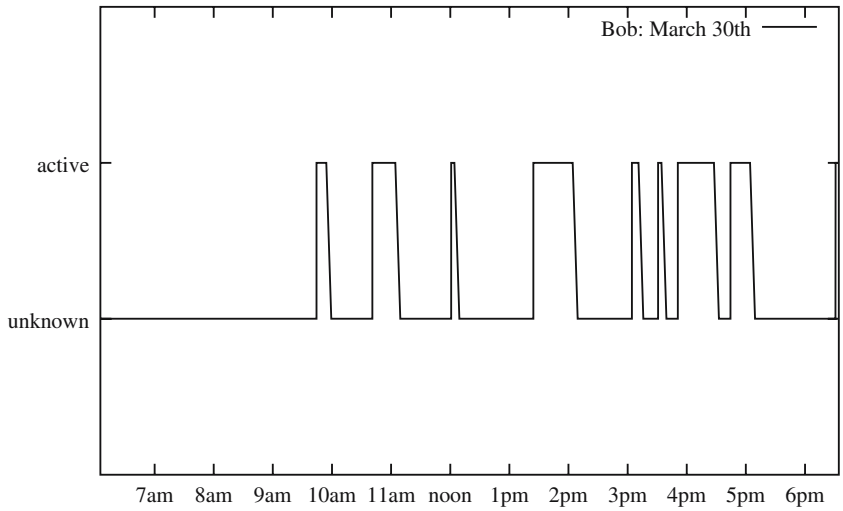

Figure 9. Display of user activity, following a user policy that they are active if they have used their keyboard or mouse in the last 5 minutes. Note that when the user is not determined to be active, we cannot conclude whether they are really active or not. So the resolved value of the activity component in this case is *unknown*.

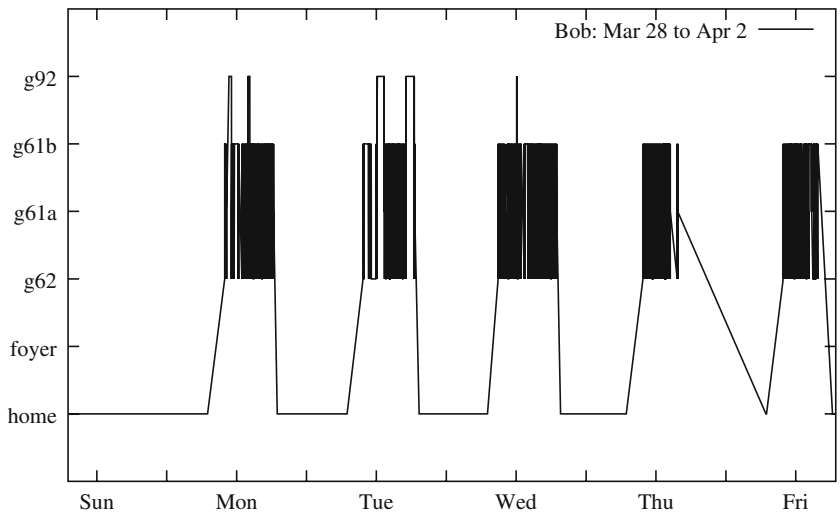

Figure 10. Bluetooth-sourced evidence for Bob over six days.

These two graphs indicate the difference in the response to requests about location that would be given to different classes of users. If Bob's child came to his door, they might be informed of his location as shown in Figure 10. In the case of typical people coming to Bob's door, the `Locator` system would first ask his user model for his availability. We now discuss the issues listed at the beginning of this section.

5.1. FLEXIBLE MODELLING OF SPATIAL ACCURACY OF LOCATION

The granularity of location modelling illustrated in Figure 8 is partly adequate for the first option in the `Locator` scenario. It does determine if Bob is near enough to come to his door in a reasonable amount of time. However, it is insufficient to identify the exact room among g61b, g61a and g62 as required in that case. This is a problem associated with overlapping Bluetooth sensors and the limited accuracy they offer. The `accretion-resolution` representation operates properly within those constraints. The approach in Figures 8 and 11 would deal properly with the second variant of the `Locator` scenario, where Bob is actually at home and the user (Natasha in the scenario) is to be told that he is unavailable.

The way that we model different degrees of spatial accuracy is by using different `resolvers`. Figure 7 was used to show the location evidence for Bob. If we use a `resolver` which simply determines a user's location on the basis of the last piece of evidence available, a continuous series of queries on Bob's location would give just this graph. Figure 8 illustrates the values returned by a `resolver` which is able to operate on the same location evidence but, in this case, the `resolver` can return only two possible values, either *work* or *unknown*. Figure 11 indicates how other policies can be used to determine the answer to requests for Bob's availability, which might be all that would be returned on a query about his location. Bob can control the degree of spatial accuracy and nature of the response available to an application that requests his location. He would do this, using a suitable interface, to select the `resolver` to be used by that application.

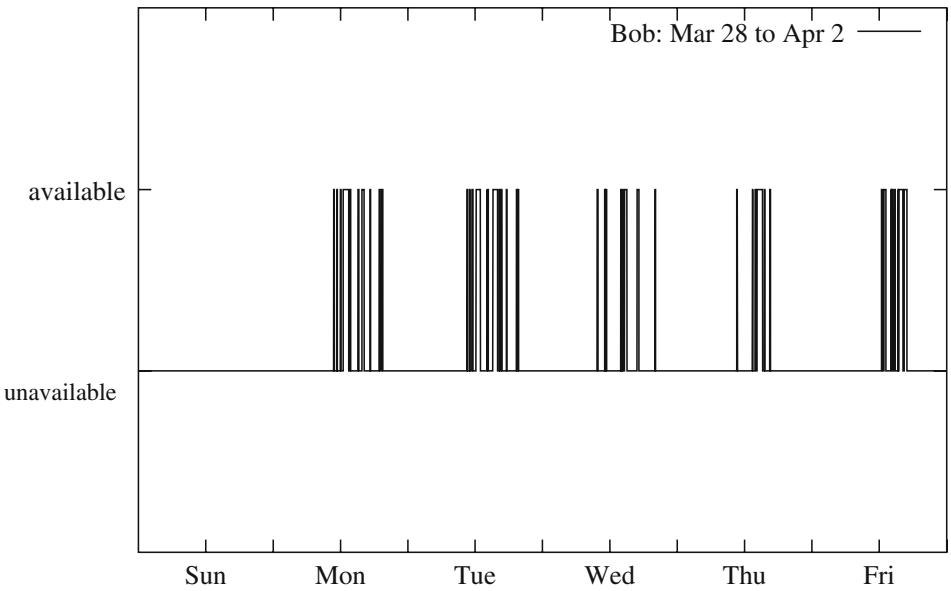

Figure 11. Availability of the user over the same six day period, following a policy that he is available when he is around the area of his office but not active at his terminal.

5.2. FLEXIBLE MODELLING OF TIME ACCURACY

This aspect operates in a similar manner to the spatial accuracy. As we saw in Figure 9, the user's activity level was determined on the basis of a `resolver` which used evidence from the last 5 minutes to determine if Bob was *active*. The indication of availability, shown Figure 11, is based upon a `resolver` which interprets evidence about the user with two sets of time granularity: the last 10 minutes in the case of the Bluetooth location sensors and 5 minutes in the case of the activity sensors.

5.3. SELECTIVE PRESENTATION OF LOCATION DEPENDING UPON WHO ASKS

Both the scenarios at the beginning of the paper indicated cases where it is desirable to control what information is available about a user, depending upon who asks. As discussed above, this can operate in terms of the spatial and temporal accuracy. Importantly, these cases also illustrated examples of the important cases where Bob's location is to be reported as *unavailable*.

6. Large Scale Experiments

In the last section, we reported detailed experiments, based upon authentic data for a single user with heterogenous evidence sources. We now report experiments on a much larger scale, involving large numbers of users, over several months.

We wanted to test with realistic data, with the potential to produce conflicting information about a person. We also wanted to experiment with data that would produce large amounts of user modelling evidence and where we choose to simulate the effects of bursts of activity. It was also desirable to have a range of users, in terms of the scalability; at one extreme, there should be some people with modest amounts of data feeding into their user models and, at the other extreme, there should be users with large amounts of evidence added to their user models.

We have found a suitable user modelling evidence source in data about logins across our computing laboratories. This produces one datum for each time an account is logged in or out of at a terminal in those labs, as well as evidence from periodic scans which survey active users each 15 minutes. These locations are spread across many rooms, some being several hundred metres apart. An individual can only be physically present in one place at a time; so any evidence indicating that an individual is in multiple locations produces conflict. Moreover, there are many ways that such conflicting evidence can occur. For example, a faulty machine might be used to log in but then the person may move elsewhere, leaving that login active while they are actually in a different room.

The full data set has over 450,000 records, covering the period 16th May to 10 September, 2004. The included periods during semester as well as examinations and breaks. There were 2496 different accounts making use of 487 terminals. Most

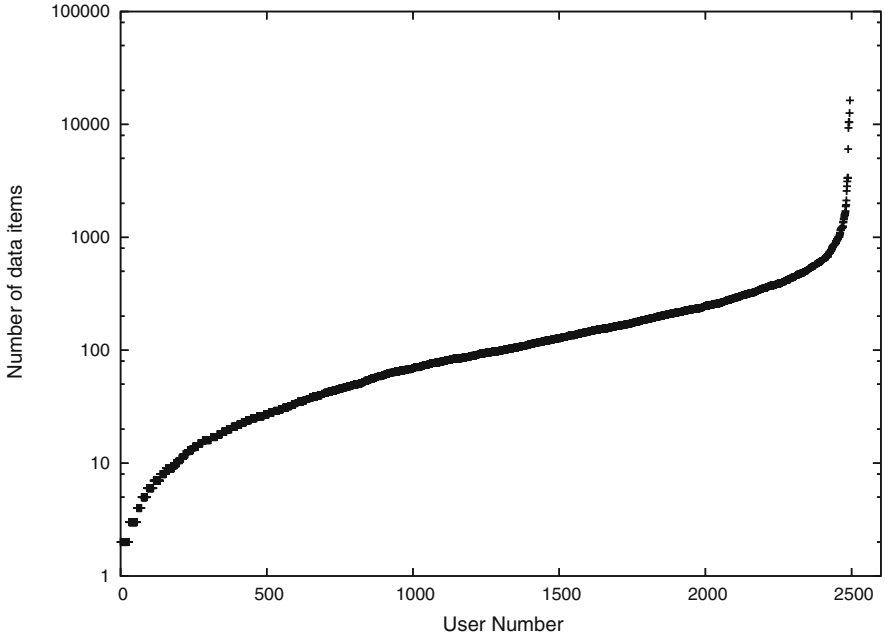

Figure 12. Overview of the amount of data per account. For the centre of the curve, it shows a gently slopping line for the log of the size of the data set against the user rank number. For the boundaries, it shows a small number of accounts with very little data and a small number with very large data sets.

accounts were for individuals, although there were some group accounts. Figure 12 shows how many pieces of evidence were collected for each account. Note the log scale on the vertical axis. The distribution is unsurprising (Zipf, 1949): there is a small number of accounts with very little evidence; small numbers having very large amounts and most forming a straight line for the log of the data set size against the rank number. Some accounts are actually used by groups of people, who, together, account for over 10,000 pieces of evidence, through the four months. This data was stored so that we could run repeated simulations for measurements under different conditions: for example, we could simulate the system being under heavy load due to a large amount of incoming sensor data.

In the remainder of this section, we describe the results of three main evaluations, exploring the scalability of the following operations as a function of the amount of evidence in the user model:

- a *tell* operation, adding one more piece of evidence to a person's user model;
- a simple *ask* operation, making use of the simplest `resolver` that determines the user's location using the Point algorithm (Hightower and Borriello, 2004) which simply accesses the most recent piece of evidence.
- a complex *ask* operation which uses a Smooth Centroid algorithm. This takes account of the last 32 minutes of user data, requiring retrieval of all evidence

from that period and calculating a weighted location prediction, where more recent locations were weighted higher;

Tests were performed on a Intel Pentium 4 CPU 2.53 GHz with 1 GB RAM.

6.1. SCALABILITY OF *tell* OPERATIONS

Figure 13 shows the CPU time for a *tell* operation. The graph shows the time taken for insertion of each additional piece of evidence is constant, giving linear scalability. The time for each *tell* is under 2 milliseconds.

6.2. SCALABILITY OF SIMPLE *ask* OPERATIONS

This test was run after each 25,000 pieces of evidence had been added to the model. This gave 19 test points, including that at the start. At each of these, we ran 400 simple *ask*s, twenty for each of twenty accounts. For these tests, we selected these accounts, in four equal-sized groups, as follows:

- accounts with the largest amount of evidence. These were three constant users, who left machines continuously logged in and group accounts with 12,603–16,329 pieces of evidence.

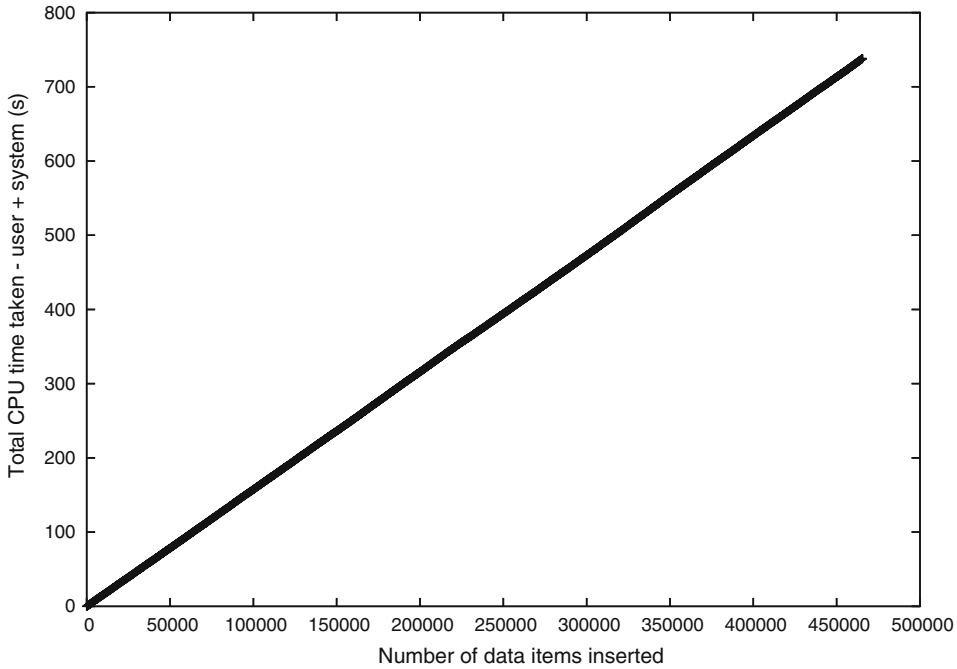

Figure 13. Time taken for *tell* operations.

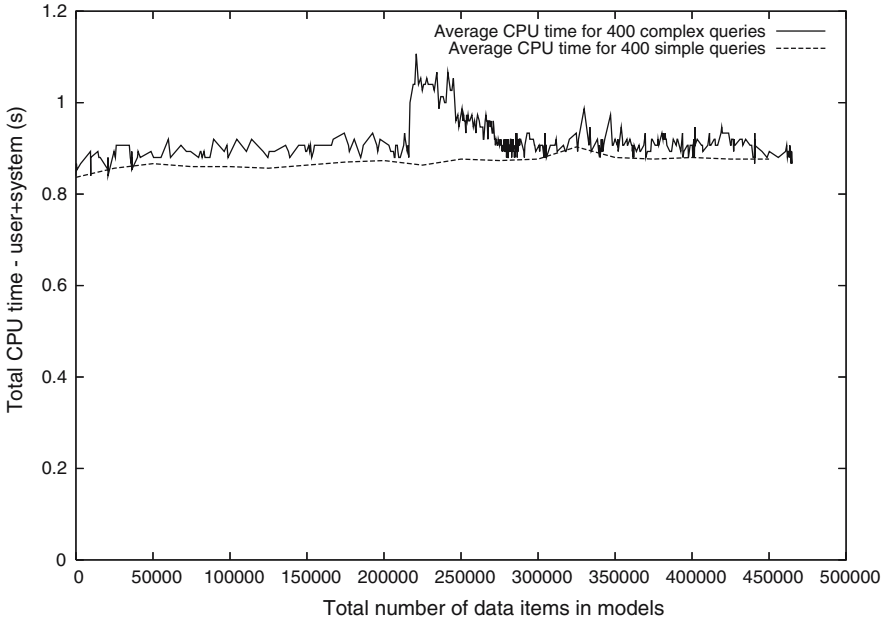

Figure 14. Performance of *ask* with complex `resolver`, compared with simple `resolver`.

- heavy individual users with 999–1309 evidence items, reflecting frequent logins over several hours each week;
- at the median of the remaining accounts were the occasional users, with around 66–67 data items, equivalent to logging into a terminal for one hour per week, such as for a single weekly tutorial;
- users who logged in once or twice in total;

The results are indicated in the broken line in Figure 14. This shows the CPU time, averaged over three runs. This indicates the time for a simple mechanism such as that of the common Point query. As the size of the model grows, this is essentially constant. It increases very slightly, but it generally took under 0.9 seconds of CPU time to complete the 400 *ask*s performed at each point, or 2.25 milliseconds per query.

6.3. SCALABILITY OF COMPLEX *ask* OPERATIONS

The second method of query was more complex. This makes use of a `resolver` which examines all data gathered about a user in the last 32 minutes. The choice of this time is based on the fact that the sensor which performs the periodic login scan runs every fifteen minutes. This means that even if the sensor is delayed slightly by other activity on the system, or network unavailabilities, it will have published at least twice in that time.

The time taken for this `resolver` to run depends on the amount of data on that account in the previous half hour. To ensure we simulated queries for all times of day, we ran at intervals of 6 hours and 35 minutes; as this does not divide into 24 hours evenly, it ensures that tests are run at all times of day.

The solid line in Figure 14 shows how this `resolver` performed compared with the simple `resolver` (broken line). In general, it takes a little more CPU time than the simple `resolver`, but it is essentially the same time cost.

The spike visible between 220,000 and 275,000 merits a little further investigation. The data stored contains details of activity for both ordinary user accounts and group accounts used by all the students in specific courses. Figure 15 shows the time for queries broken down into account types. The solid line is for the group accounts. The spike at around 250,000 evidence items is due to the group accounts and is caused by one account being used continuously by all 100 students doing an intensive course over about a week. For this account, each *ask* means that the `resolver` must examine every piece of data for the given account which has been collected within the last 32 minutes. This requires examination of 300–400 pieces of data at times during this week. The peak of this is somewhat over 1 second.

For 99.9% of *ask*s, the time taken was less than 0.02 seconds for the five *ask*s, typically 4 milliseconds per *ask*. In most cases, the time taken was half this, at 2 milliseconds per *ask*.

In Section 4, we described the process involved in determining a user's location, based upon detections of a mobile device. We now link this to the scalability experiments just reported. Based on these experiments, each *ask* and *tell* on the user, device and sensor models incurs a similar cost, of approximately 2 milliseconds. If we expected that many applications would require this information, we could easily improve the efficiency of this location calculation. For example, a continuous process could perform this calculation at regular intervals, say, every 5 minutes. Then, each of the applications would use a `resolver` that made use of this value, reducing the cost to each application to a constant.

Location requests need to be answered within reasonable limits of human patience. To assess this, we follow (Nielsen, 2000) who, in turn cites classic work, indicating that response time up to 0.1 seconds appears instantaneous, up to 1.0 second gives continuity in the interaction, while any more than 10 seconds causes the user to shift their attention from the current dialogue. This means that a natural interaction with an invisible and unobtrusive environment requires that location be determined near the 0.1 second speed. On the other hand, in cases where the user would naturally expect to wait, such as in our `Locator` scenario, times up toward 10 seconds might be acceptable.

We can now reflect on how the experiments just described relate to the response time for an application, such as those in the scenarios at the start of the paper. Our evaluations have been restricted to the pure assessment of the `accretion-resolution` representation since that is the focus of this paper.

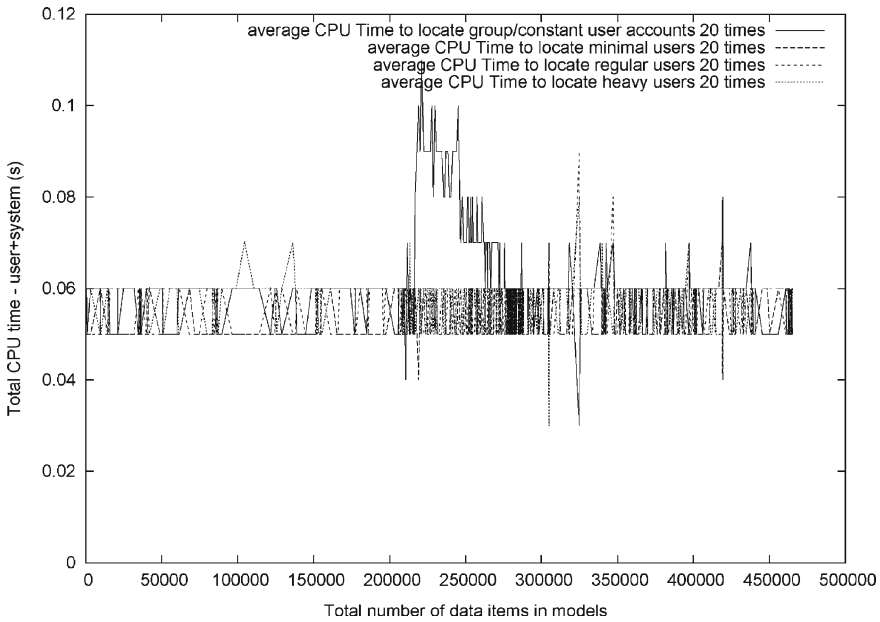

Figure 15. Time for queries broken down into account types.

There are many other aspects of scalability in the architecture. For example, the Publish/Subscribe server has latency associated with it. There are also some serious issues associated with the sensor technologies we have used: in particular, the Bluetooth sensors take of the order of 10 seconds to detect a user. There are also some serious, and potentially difficult to predict, latencies in the networks supporting a complete ubiquitous computing application. For example, the system described in Section 3 involves a sensor at Bob's home. This sends a message from a machine at his home to the machine that maintains the sensor model: in our current implementation, this is at Bob's workplace. The delivery program is able to cope well with arbitrary network difficulties, such as noise and intermittent failures. If there are such problems, the message could be delayed. Moreover, as was illustrated in Figure 4, it may require many *ask*s on device and sensor models to collect the evidence for a single location determination. Also, as described in Section 3.5, the architecture of our current version of `MyPlace` involves many different machines, each with potentially varying loads affecting their response times and with several network connections each with varying latencies.

In summary, to determine the actual response time for a query is strongly dependent upon the detailed architecture of the ubiquitous computing application. Clearly, it is important to engineer this to match the needs of the application. However, from the experiments reported above, we can conclude that the overhead of queries on the sensor, device and user models that make use of our `accretion-resolution` representation is very modest and well within the timescales for responses.

6.4. EVALUATION OF CONFLICT MANAGEMENT

We now report experiments into the nature of the conflicting evidence about user's locations, based upon the login sensor data. Although this analysis is restricted to a single sensor, we will use this as a basis for describing how the `accretion-resolution` approach tackles the problems involved in reasoning about aspects like user location from such unobtrusive sensors.

For each account, we defined a *login event* to include each login, and a *scan event* as each detection of the logged-in account during the periodic scans. For each of these events, we identified all evidence indicating any conflicts in the user's current location. Essentially, we maintained a list of all locations for which there was evidence from one of the login or scan events. If this list had any conflicts, we concluded that the event showed inconsistency.

The calculation of inconsistency had to take account of the fact that people can move. To do this at each event, as described above, we compiled all evidence of events from that scan period, which means all evidence within the last 15 minutes under normal operation. (There were sensor restarts approximately once per day and these altered the timing slightly. But the effect was small.) We use this to create a list of all the locations that are suggested by the available evidence. So, for example, if the scan indicates the user is at two terminals which are in different locations, this list has those two locations. In addition to this, we make use of data about logouts to indicate that the user left that terminal: we use this information to remove that location from the list of possible user locations. This way if a user logs out from one terminal, walks down the hall and logs into a different terminal, this is not treated as a conflict. (A more accurate measure would take account of the time to move between the logout and login location.)

The actual calculation examines the list of location evidence associated with the current event. It identifies the location that has the most evidence. We then calculate the percentage of evidence for this location over the total amount of evidence on the list. So, for example, if the location list has 10 pieces of evidence, of which five are for Location A and five for Location B, this gives a 50% consistency value.

Figure 16 shows the results. The horizontal axis is the account rank, as in Figure 12. The figure has one cross for each account, indicating the consistency as just described. So, for example, at rank 1764, we can see an account which has 136 events. The analysis indicated that in 74 cases, there was no conflict and the remaining 62 had conflict. Studying the detailed evidence, it emerges that this person logged in for 5 minutes, then logged out, returning one week later, logging into 2 terminals in nearby rooms, within a few minutes, then leaving these logged in overnight.

The figure is dominated by the solid bar near 100%, indicating that, for most accounts, the consistency was very high. To gain a sense of the nature of the conflicting evidence, we studied the data for those accounts with the highest proportion

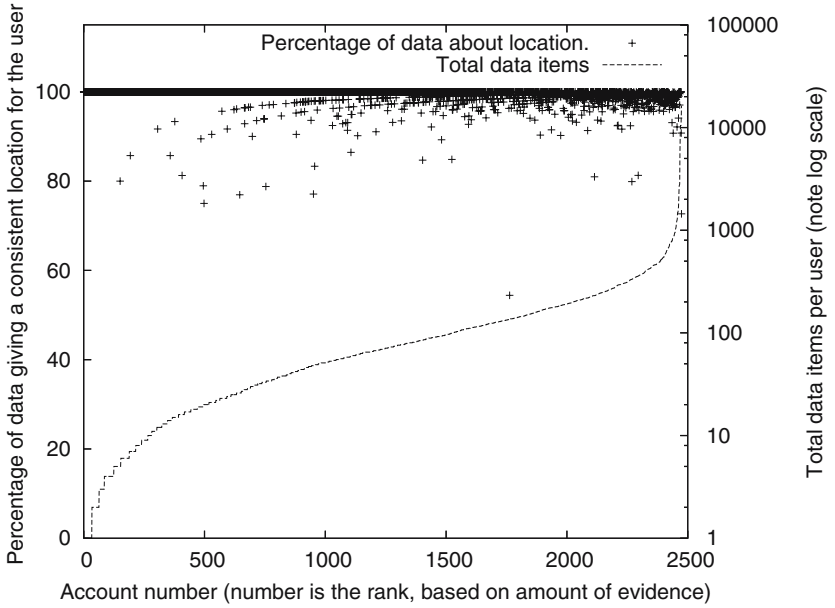

Figure 16. The left axis shows the percentage of *login events* and *scan events* with all evidence consistent, compared with the total number of events. Each cross indicates this percentage for one account. The right axis shows the log of the number of events and the S-shaped curve shows the number of events per account.

of events with conflicting evidence. The reasons for the conflict can be categorised as follows:

1. group or shared accounts;
2. a terminal is normally left logged in for very long periods, for example, in the person's office;
3. a person fails to log off properly, for example, due to a faulty machine which breaks leaving the person logged in while they move to another machine;
4. account sharing, with different people commonly using the same account in different locations;
5. very small data collections, where even modest effects from cases 3 and 4 above can give large proportions of inconsistent data.

We can ignore the first if we limit user modelling to evidence from sensor data for the individual. The others describe inherent limitations in this class of evidence source. Notably, correct management of these problems will differ for different people; for example, the second will be relevant only to people with an office.

This evaluation explored the degree and nature of inconsistency due to the unobtrusive login sensor. The `accretion-resolution` representation provides the framework for collecting and resolving such conflicting evidence. There are many algorithms that can be applied, within different `resolvers`, to deal with

conflict, as discussed in the section on `resolvers` in Section 3.3. These can incorporate the sensor characteristics and additional evidence from other sensors. Notably, it seems likely that different people may elect to personalise the choice of resolver that is applied to them.

7. Related Work

An excellent review (Kobsa, 2001a) indicates the breadth of representations used by generic user modelling shells. He identified a trend towards lighter weight, simpler representations. Early systems tended to come from work in artificial intelligence and natural language understanding and these valued generality, expressiveness and powerful inferences in representations. For example, UMT (User Modelling Tool) (Brajnik and Tasso, 1994) had a database of all the user models held by the system, a knowledge base of stereotypes (Rich and Wahlster, 1989) in a multiple inheritance hierarchy, the database of possible user models for the current user, a rule-base of constraints on values of attributes in the user model and inference rules for generating new user modelling information as well as a consistency manager based on an truth maintenance based approach (Doyle, 1979) and associated mechanisms. Similarly, BGP-MS models user's Beliefs, Goals and Plans (Kobsa and Pohl, 1995), representing *concepts* in the user model as an inheritance hierarchy. Each concept was described by a four-tuple: a role predicate for each relation this concept participates in; value restrictions on the arguments of each relation; a number of restrictions indicating how many of the attributes were required for an instance of this concept; a modality to indicate whether an attribute was necessary or not. Such concepts were kept in *partitions* which themselves could be organised into inheritance hierarchies. These provided a representation for alternate views of knowledge, such as SB, the system's beliefs, SB(UB), the system's beliefs about the user's beliefs, SB(UB(SB)), the system's beliefs about the user's beliefs about the system's beliefs.

One of the important representational elements in user modelling is the stereotype, a term coined by Rich (1989) to mean '*a collection of attributes that often co-occur in people ... they enable the system to make a large number of plausible inferences on the basis of a substantially smaller number of observations. These inferences must, however, be treated as defaults, which can be overridden by specific observations*' (Rich, 1989: 35). They have been explicitly incorporated into the representations of UMT and BGP-MS as well as several other systems which explored general representations for user modelling, for example: Generalised User Modelling System, GUMS (Kass, 1991); TAGUS (Paiva and Self, 1995) which also supports inferences about the user's reasoning about knowledge; the Student Modelling Maintenance System, SMMS, (Huang et al., 1991) where the latter two had support for truth maintenance.

These early systems dealt with the challenges of uncertainty and inconsistency with truth maintenance based approaches (Doyle, 1979). These are rather

heavy-weight for a ubiquitous environment. THEMIS (Kono et al., 1994) explored the role of inconsistency and when it needs to be resolved by the system. Notably, it highlighted the possibility that people may have inconsistent beliefs: a system which tries to construct a consistent model of them imposes constraints that are inappropriate. In the case of ubiquitous computing, a very important source of inconsistency in the user model will be due to the unreliability of information from sensors. More importantly, many aspects of the user, including their location, will change frequently. A representation for ubiquitous computing must deal with this efficiently.

An important form of stereotype is the double stereotype (Chin and Wahlster, 1989) which enables reasoning in two directions. For example, information about the user could be used in stereotypic inference about their predicted behaviour; in the other direction, limited sensor observations of their behaviour can be used to infer other attributes about them. Taking a concrete example, suppose that an active-badge tracking system *observes* a user in the executive coffee suite on a few occasions. This could be used to infer that this person is an executive. Another person who has just joined the organisation as an executive could be predicted to visit the executive coffee suite.

More recent work has been characterised by simpler representations, matching the needs of emerging personalised applications such as recommenders and personalised web sites. It seems that quite modest user models have been useful for recent ubiquitous computing systems, such as (Hatala and Wakkary, 2005; Petrelli and Not, 2005). As Kobsa observes, the demands of personalised applications will determine the characteristics which should be provided by a user model representation. Our `accretion-resolution` representation is in the spirit of such simpler representations but is distinctive in differentiating evidence sources so that these can play a role in the general operation of *resolver* algorithms and in the selective and tailored granularity of *ask*s that each application is allowed.

Another dimension of the representation of user models relates to the ontology which defines the semantics of the user modelling language. Heckmann has used an XML-based user modelling language (Heckmann, 2003b) for Ubi's world, and has provided a user interface to define policies for access to the user model (Heckmann, 2003a). Heckmann also has provided an interface (Heckmann, 2004) that enables users to scrutinise the representation as well as the semantics defining them.

An important design goal for the `accretion-resolution` representation has been that the representation should support user scrutiny and control of personalisation: both in terms of being able to access and control the *values* represented in the user model and the *processes* that defined them. It is widely acknowledged that there are critical privacy and control issues for ubiquitous computing environments. For example, from the early work on active badges (Want et al., 1992) to recent work based on similar approaches (Priyantha et al., 2000), the researchers noted concerns about privacy. In such work, an important basis for

managing privacy is based on the fact that devices, rather than people are tracked. Our architecture explicitly supports this.

It was also observed that *it is the way location data is used or logged that is often the main concern for badge wearers*. In all this work, including more recent research on privacy of location information (Hengartner and Steenkiste, 2003), the focus has been on the security of the information without proposing that a user model would be a natural repository for the information and its management. The work reported in this paper represents an important departure from that dominant view of location information.

Our work is also complementary to the research into policy languages and specifications for managing privacy (Godefroid et al., 2000) as well as middleware for policy management, such as LocServ (Myles et al., 2003) and the pawS, privacy awareness system for ubquitous computing environments (Langheinrich, 2002b). These have been influenced by P3P (P3P, 2004) which was developed in the context of web privacy. There is also the very important need for interfaces enabling users to define privacy policies (Lau et al., 1999; Lederer et al., 2003) as well as (Heckmann, 2003a) mentioned above. Another loosely related area of work concerns anonymity and pseudonymity, which has been linked to user modelling (Kobsa and Schreck, 2003). There has also been considerable work on design guidelines for ubiquitous computing systems, such as (Bellotti et al., 2002), and the principles of design for privacy awareness in ubiquitous system (Langheinrich, 2001). He further argues (Langheinrich, 2002a) that privacy requirements should include *comprehensibility* and *manageability*, *control* and *accountability* for the use and management of personal data collected in the environment. A complementary concern for these issues has been described in the context of user modelling and personalisation (Kobsa, 2002; Miller, 2000). Some work towards addressing these issues has been done. For example, *privacy mirrors* (Mynatt and Nguyen, 2001) enable users to see what sensors are detecting them and how information about them has been used. This is somewhat similar to our invisibility interface.

The `accretion-resolution` representation and the `MyPlace` architecture match well the ongoing refinements to a range of national and international privacy legislation (Kobsa, 2001b). For example, the European Community Directive on Data Protection (EU and the Council of the European Union, 1995) mandates 'right of access to and the right to rectify the data' and, a particular demand for scrutability in the requirement for 'an intelligible form of the data undergoing processing and of any available information as to their source'.

8. Discussion and Conclusions

The use of *accretion* and *resolvers* provides an elegant and flexible way to deal with inconsistency. The whole point of modelling some components is to track changes. For example, the user's location changes as they move around their environment.

There is no need to deal with conflicting evidence until an application requests the value of components of the user model. Until a value is required, the `accretion-resolution` representation allows evidence to *accrete* in the models of sensors and devices without interpreting it. When an application needs to know the user's location, it can *ask* for the current *resolve*d value of the evidence available for that component. If it is authorised to have that information, then at that point, the `resolver` it has nominated is used to determine the value of the component. In the case of the user's location, the values returned by this request would be either an indication that the system really does not know, or the value of the modelled location, possibly with a certainty value.

A primary motivation for the design of `accretion-resolution` was to provide a *scrutable* representation, where the user could examine the user model, the *value* of its components such as inferred location and the *processes* that define that value. Those processes can be explained in terms of the list of evidence and the `resolver` process used to interpret it. We see scrutability as a foundation for *user control* over personalisation, an issue that is also explored by (Cheverst et al., 2005). In this paper, we have described how the design of the `accretion-resolution` representation supports modelling of users in a ubiquitous computing environment, in a manner that should be amenable to user scrutiny. Discussion of the very important issues of the design of suitable scutability interfaces is beyond the scope of this paper but has been explored in previous work (Apted et al., 2003; Czarkowski and Kay, 2003; Kay, 1995; Uther and Kay, 2003) including some exploration of ubiquitous computing environments (Kay et al., 2003).

This introduces another dimension of a practical deployment of `MyPlace`: we need to explore ways that people can select a suitable policy for the ways that applications should be allowed to access `accretion-resolution` user models. One natural possibility would make use of visualisations such as the graphs in Section 5: these would enable a person to see how a policy would operate in practice.

We have barely touched upon the underlying modelling of space: the Invisibility Scenario requires this. We have chosen to model locations at a symbolic level, in terms of rooms and other places that make sense to people. This matches proposals (Hightower and Borriello, 2001; Lin et al., 2002) which call for intuitive modelling of location. We have also performed some preliminary experiments towards interfaces that enable people to personalise the ontology of the underlying space models used for them (Carmichael et al., 2004). Further discussion of this important topic is outside the scope of this paper. There are other issues that are very important but were outside the scope of this paper. For example, we have not addressed security and privacy. Nor have we explored the details of the engineering of the actual machines, networks and their configurations.

We have illustrated how `MyPlace` demonstrates the effectiveness of the `accretion-resolution` user model representation for an interesting class of problems for user modelling in a ubiquitous computing environment. We began this paper

with two scenarios, one illustrating the problems of invisibility in ubiquitous computing environments and the second, `Locator`, illustrating the need for flexible spatial and temporal accuracy in the modelling of location and the need to be able to efficiently determine a user's location and ensure that it can be reported differently for different users. We have described how the `accretion-resolution` representation addresses these issues.

This work makes the following contributions.

- It introduces the `MyPlace` architecture, which derives considerable power and flexibility from the use of a *Publish/Subscribe* server to receive and redirect the messages from sensors to sensor and device models. This is an inherently distributed approach that copes naturally with geographically spread sensors. This approach has not been explored in previous work on personalised ubiquitous computing environments.
- `MyPlace` makes use of the redundant Seen and Seenby components, a novel approach, that serves to separate the elements of the system. This is important as a foundation for controlling access to information for privacy and security as well as for partitioning information within the personalised ubiquitous computing environment.
- We have illustrated how `MyPlace` manages multiple, heterogenous sensors in a light-weight approach designed to be simple enough to support scrutability as well as user controlled, flexible interpretation of available evidence via *resolvers* that could, potentially, be chosen and tuned by users.
- The `accretion-resolution` representation was implemented as a light-weight tool and we have demonstrated the efficiency of the core *tell* and *ask* operations.
- Finally, we have extended the use of a user model representation to model, not only users, but also the other entities in the personalised ubiquitous computing environment, the sensors and devices. This provides a consistent representation and reasoning mechanism across these three classes information and recognises that the same core reasoning flexibility can be useful for all of these. This provides an elegant mechanism to address the need to integrate context modelling and user modelling (Zimmermann et al., 2005).

We have illustrated these contributions in terms of the two demonstrator `MyPlace` applications. The first addresses the *invisibility* scenario. This is a novel but potentially important personalised ubiquitous computing application, which has much in common with more established work that delivers information based upon the user's location (Hatala and Wakkary, 2005; Petrelli and Not, 2005). The second scenario, `Locator`, goes beyond the many systems that have used various sensor technologies to model people's location, exploring the flexible personalisation of modelling people's location and activity.

Appendix A

Annotation	Actual data collected from sensors format: location of sensor, time
Starts at home	'Home', Tue Mar 30 06:54:38 2004 'Home', Tue Mar 30 06:58:46 2004
Seen in the building foyer	'Madsen Foyer', Tue Mar 30 09:40:13 2004
Offices near Bob's office (G61a)	'G62', Tue Mar 30 09:41:55 2004 'G61a', Tue Mar 30 09:42:02 2004 'G61b', Tue Mar 30 09:42:04 2004
Activity detected on Bob's system	'G61a system', Tue Mar 30 09:44:17 2004
Note very close or overlapping detections	'G61a', Tue Mar 30 09:49:12 2004 'G61b', Tue Mar 30 09:49:13 2004 'G62', Tue Mar 30 09:50:06 2004
System activity disambiguates	'G61a system', Tue Mar 30 09:50:57 2004 'G62', Tue Mar 30 09:51:07 2004 'G61a', Tue Mar 30 12:05:07 2004 'G61b', Tue Mar 30 12:05:08 2004
Seminar/lunch	'G92', Tue Mar 30 12:10:04 2004 'G61a', Tue Mar 30 12:10:04 2004 'G92', Tue Mar 30 13:16:27 2004
Brief visit back to office	'G61b', Tue Mar 30 13:18:43 2004 'G92', Tue Mar 30 13:19:31 2004 'G61a', Tue Mar 30 13:20:45 2004 'G61a system', Tue Mar 30 13:24:18 2004 'G61a', Tue Mar 30 18:34:32 2004
Leaving	'Madsen Foyer', Tue Mar 30 18:34:42 2004
At home	'Home', Tue Mar 30 19:20:11 2004

Annotated subset of data collected for a single user on Tuesday 30th March 2004. There were 336 sensor entries for this user. Annotations indicate main locations and conflicting sensor evidence. Blank lines indicate that data has been omitted at this point so that we can show those entries that are important for the discussion. The ruled line indicates the end of the data used for Table I.

Acknowledgements

This research was partially funded by the Smart Internet Technology Cooperative Research Centre Intelligent Environments Programme. We thank Chris Johnson, Australian National University, for his detailed and very helpful comments on the earlier draft of this paper and encouragement for this project.

References

Ackerman, M., Cranor, L. and Reagle, J.: 1999, Privacy critics: UI components to safeguard user' privacy'. In: M.H. Williams (ed.): *The CHI is the Limit: Human Factors in Computing Systems, CHI99 Conference Proceedings*. Pittsburgh, Pennsylvania, USA, pp. 1–8.

Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A. and Hopper, A.: 2001, Implementing a sentient computing system. *IEEE Computer* **34**(8), 50–56.

Apted, T., Kay, J., Lum, A. and Uther, J.: 2003, Visualisation of ontological inferences for user control of personal web agents. In: E. Banissi, K. Borner, C. Chen, G. Clapworthy, C. Maple, A. Lobben, C. Moore, J. Roberts, A. Ursyn, and J. Zhang (eds.): *Proceedings of IV03-VSW, Information Visualisation – Semantic Web Visualisation*. London, UK, pp. 305 – 311.

Bauer, M., Becker, C. and Rothermel, K.: 2002, Location models from the perspective of context-aware applications and mobile ad hoc networks. *Personal and Ubiquitous Computing* **6**(5-6), 322–328.

Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A. and Lopes, C.V.: 2002, Making sense of sensing systems: five questions for designers and researchers. In: *CHI '02: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Minneapolis, Minnesota, USA, pp. 415–422.

Brajnik, G. and Tasso, C.: 1994, A shell for developing non-monotonic user modeling systems'. *International Journal of Human-Computer Studies* **40**(1), 36–62.

Brummit, B., Krumm, J., Kern, A. and Shafer, S.A.: 2000, Easyliving: technologies for intelligent environments'. In: *Proceedings of: 2nd International Symposium on Handheld and Ubiquitous Computing*. Bristol, United Kingdom, pp. 12–29.

Carmichael, D.J., Kay, J. and Kummerfeld, R.J.: 2004, Personal ontologies for feature selection in intelligent environment visualisations. In: J. Baus, C. Kray, and R. Porzel (eds.): *Proceedings of AIMS'04 – Artificial Intelligence in Mobile System*. Nottingham, England, pp. 44–51.

Cheverst, K., Byun, H.E., Fitton, D., Sas, C., Kray, C. and Villar, N.: 2005, Exploring issues of user model transparency and proactive behaviour in an office environment control system. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* (this issue).

Chin, D. and Wahlster, W.: 1989, KNOME: modeling what the user knows in UC. In: A. Kobsa (ed.): *User Models in Dialog Systems*. Springer-Verlag, Berlin, pp. 74–107.

Czarkowski, M. and Kay, J.: 2003, How to give the user a sense of control over the personalization of adaptive hypertext?. In: P. de Bra, H. Davis, and J. Kay (eds.): *Proceedings of Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, User Modeling 2003 Session*. University of Pittsburgh, Johnstown, *PA*, USA, pp. 121–132.

Doyle, J.: 1979, A truth maintenance system. *Artificial Intelligence* **12**(3), 231–171.

EU, T.E.P. and the Council of the European Union, 1995, European Community Directive on Data Protection. http://www.doc.gov/ecommerce/eudir.htm.

Godefroid, P., Herbsleb, J.D., Jagadeesan, L.J. and Li, D.: 2000, Ensuring privacy in presence awareness systems: an automated verification approach. In: *Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*. Philadelphia, Pennsylvania, USA.

Hatala, M. and Wakkary, R.: 2005, User modeling and semantic technologies in support of a tangible interface. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* (this issue).

Heckmann, D.: 2003a, Integrating privacy aspects into ubiquitous computing: a basic user interface for personalization. In: *Proceedings of Artificial Intelligence in Mobile Systems Workshop at UbiComp 2003*, Seattle, Washington, USA.

Heckmann, D.: 2003b, A specialised representation for ubiquitous computing and user modeling. In: *Proceedings of Workshop on User Modelling for Ubiquitous Computing at User Modelling 2003 (UM'03)*. University of Pittsburgh, Johnstown, PA, USA. http://www.di.uniba.it/ ubium03/heckmann-2.pdf.

Heckmann, D.: 2004, Ubisworld. http://www.u2m.org/.

Hengartner, U. and Steenkiste, P.: 2003, Protecting access to people location information. In: *Proceedings of First International Conference on Security in Pervasive Computing*. Boppard, Germany, pp. 25–38.

Hightower, J. and Borriello, G.: 2001, Location systems for ubiquitous computing. *IEEE Computer* **34**(8), 57–66.

Hightower, J. and Borriello, G.: 2004, Particle filters for location estimation in ubiquitous computing. In: N. Davies, E. Mynatt, and I. Siio (eds.): *Proceedings of 6th International Conference on Ubiquitous Computing (UbiComp 2004)*. Nottingham, England, pp. 88–106.

Huang, X., McCalla, G.I., Greer, J.E. and Neufeld, E.: 1991, Revising deductive knowledge and stereotypical knowledge in a student model. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **1**(1), 87–116.

Judd, G. and Steenkiste, P.: 2003, Providing contextual information to pervasive computing applications. In: *Proccedings of IEEE International Conference on Pervasive Computing (PERCOM)*, Dallas, pp. 133–142.

Kass, R.: 1991, Building a user model implicitly from a cooperative advisory dialog. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **1**(3), 203–258.

Kay, J.: 1995, The um toolkit for cooperative user modelling. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **4**(3), 149–196.

Kay, J., Kummerfeld, B. and Lauder, P.: 2002, Personis: a server for user models. In: P.D. Bra, P. Brusilovsky, and R. Conejo (eds.): *Proceedings of AH'2002, Adaptive Hypertext 2002*. Malaga, Spain, pp. 203–212.

Kay, J., Lum, A. and Uther, J.: 2003, How can users edit and control their models in ubiquitous computing environments?. In: K. Cheverst, N. de Carolis, and A. Kruger (eds.): *UM03 Workshop on User Modeling for Ubiquitous Computing*. University of Pittsburgh, Johnstown, PA, USA, pp. 12–16.

Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E.D., Starner, T.E. and Newstetter, W.: 1999, The Aware Home: a living laboratory for ubiquitous computing research. In: *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, Pittsburgh, USA, pp. 191–198.

Kobsa, A.: 2001a, Generic user modeling systems. *User Modeling and User-Adapted Interaction – Ten Year Anniversary Issue* **11**(1–2), 49–63.

Kobsa, A.: 2001b, Invited keynote: tailoring privacy to user's needs. In: *Proceedings of the User Modeling Conference*. Sonthofen, Germany, pp. 303–313.

Kobsa, A.: 2002, Personalized hypermedia and international privacy. *Communications of the ACM* **45**(5), 64–67.

Kobsa, A. and Pohl, W.: 1995, The user modeling shell system BGP-MS. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **4**(2), 59–106.

Kobsa, A. and Schreck, J.: 2003, Privacy through pseudonymity in user-adaptive systems. *ACM Transactions on Internet Technology* **3**(2), 149–183.

Kono, Y., Ikeda, M. and Mizoguchi, R.: 1994, THEMIS: a nonmonotonic inductive student modeling system. *International Journal of Artificial Intelligence in Education* **5**(3), 371–413.

Langheinrich, M.: 2001, Privacy by design – principles of privacy-aware ubiquitous systems. In: G.D. Abowd, B. Brumitt, and S.A. Shafer (eds.): *Proceedings of 3rd International Conference on Ubiquitous Computing (UbiComp 2001)*, Atlanta, Georgia, USA, pp. 273–291.

Langheinrich, M.: 2002a, *As we may live – real-world implications of ubiquitous computing*. http://www.vs.inf.ethz.ch/publ/papers/uc-implications.pdf.

Langheinrich, M.: 2002b, A privacy awareness system for ubiquitous computing environments. In: G. Borriello and L.E. Holmquist (eds.): *Proceedings of 4th International Conference on Ubiquitous Computing (UbiComp 2002)*, Goteberg, Sweden, pp. 237–245.

Lau, T., Etzioni, O. and Weld, D.S.: 1999, Privacy interfaces for information management. *Communications of the ACM* **42**(10), 89–94.

Lederer, S., Hong, J.I., Jiang, X., Dey, A.K., Landay, J.A. and Mankoff, J.: 2003, Towards everyday privacy for ubiquitous computing of everyday privacy in ubiquitous computing environments. Technical report ucb-csd-03-1283, Computer Science Division, University of California, Berkley.

Lehmann, O., Bauer, M., Becker, C. and Nicklas, D.: 2004, From home to world - supporting context-aware applications through world models. In: *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PERCOM'04)*. Orlando, Florida, pp. 297–306.

Lin, J., Laddaga, R. and Naito, H.: 2002, Personal location agent for communicating entities (PLACE). In: F. Paternò (ed.): *Proceedings of Fourth International Symposium on Human Computer Interaction with Mobile Devices (Mobile HCI 2002)*, Vol. 2411 of *Lecture Notes in Computer Science*. Pisa, Italy, pp. 45–59.

Miller, C.: 2000, Rules of etiquette, or how a mannerly AUI should comport itself to gain social acceptance and be perceived as gracious and well-behaved in polite society. In: S. Rogers and W. Iba (eds.): *Papers from 2000 Adaptive User Interfaces AAAI Spring Symposium, Technical Report SS-00-01*, Stanford, CA:, pp. 80–81.

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

Mynatt, E.D. and Nguyen, D.: 2001, Making Ubiquitous Computing Visible. In: *Proceedings of CHI 2001 Workshop on Building the Ubiquitous Computing User Experience*. Minneapolis, Minnesota. http://www2.parc.com/csl/ projects/ubicomp-workshop/position-papers/mynatt.pdf (visited April 2003).

Nielsen, J.: 2000, *Designing Web Usability*. Indianapolis, Indiana, USA: New Riders.

P3P: 2004, Platform for Privacy Preferences (P3P). http://www.w3.org/P3P/ (visited April 2004).

Paiva, A. and Self, J.: 1995, TAGUS – a user and learner modeling workbench. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* **4**(3), 197–228.

Patterson, D., Liao, L., Gajos, K., Coller, M., Livic, N., Olson, K., Wang, S., Fox, D. and Kautz, H.: 2004, Opportunity knocks: a system to provide cognitive assistance with transportation services. In: N. Davies, E. Mynatt, and I. Siio (eds.): *Proceedings of 6th International Conference on Ubiquitous Computing (UbiComp 2004)*. Nottingham, England, pp. 433–450.

Petrelli, D. and Not, E.: 2005, User-centred design of flexible hypermedia for a mobile guide: reflections on the HyperAudio Experience. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research* (this issue).

Priyantha, N.B., Chakraborty, A. and Balakrishnan, H.: 2000, The Cricket location-support system. In: *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, Boston, Massachusetts, pp. 32–43.

Rich, E. and Wahlster, W.: 1989, Stereotypes and user modeling. In: A. Kobsa (ed.): *User Models in Dialog Systems*. Berlin: Springer-Verlag, pp. 35–51.
Scott, J. and Hazas, M.: 2003, User-friendly surveying techniques for location-aware systems. In: *Proceedings of 5th International Conference on Ubiquitous Computing (UbiComp 2003)*. Seattle, Washington, USA, pp. 44–53.
Segall, B., Arnold, D., Boot, J., Henderson, M. and Phelps, T.: 2000, Content based routing with Elvin4. In: *Proceedings AUUG2K*. Canberra, Australia. http://elvin.dstc.edu.au/doc/papers/auug2k/auug2k.pdf.
Uther, J. and Kay, J.: 2003, A web-based visualion of large user models. In: P. Brusilovsky, A. Corbett, and F. de Rosis (eds.): *Proceedings of 9th International Conference on User Modeling (UM 2003)*. Johnston, PA, USA, pp. 198–202.
Want, R., Hopper, A., Falco, V. and Gibbons, J.: 1992, The active badge location system. *ACM Transactions on Information Systems* **10**(1).
Zimmermann, A., Sprecht, M. and Lorenz, A.: 2005, Personalisation and context-management. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*. (this issue).
Zipf, G.K.: 1949, *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA.

Authors' Vitae

David J Carmichael
David Carmichael is a Ph.D. candidate in the School of Information Technologies as the University of Sydney. He received his B.Sc. with First Class Honours at the University of Sydney. His research is supported by the Smart Internet Technology Cooperative Research Centre (SITCRC), and is in the areas of Ubiquitous Computing and Intelligent Environments. He is particularly interested in exploring the challenges of producing personalised views of the invisible elements, sensors, services and nearby people, within an Intelligent Environment.

Judy Kay
Judy Kay is Associate Professor at School of Information Technologies, at the University of Sydney. She is co-leader of the Smart Internet Technology Research Group which conducts fundamental as well as applied research in user-adapted systems. The core of her work is to represent and manage personalisation that ensures the user can maintain control, being able to scrutinise and control the whole process of personalisation. She has over 100 publications in the areas of personalisation and teaching and learning.

Bob Kummerfeld
Bob Kummerfeld is an Associate Professor of Computer Science at the University of Sydney working in the area pervasive computing. He is particularly interested in scalable message systems for the distribution and management of context information and user modelling for pervasive computing. He is co-leader of the Smart Internet Technology Research Group.