Consistent modelling of users, devices and environments in a ubiquitous computing environment †

J Kay

R J Kummerfeld

D J Carmichael

School of Information Technologies
University of Sydney, Australia 2006
{judy,bob,dcarmich}@it.usyd.edu.au

ABSTRACT

This paper describes the use of an accretion/resolution user modelling representation to model people, places and things. 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 in modelling user location and activity in such an environment, showing how our approach supports different spatial and temporal granularity in location modelling and tests of the speed of determining the values of these aspects of the user model.

Keywords: user model representation, modelling location, scrutability, user control, modelling pervasive computing environments

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

† This paper (or a similar version) is not currently under review by a journal or conference, nor will it be submitted to such within the next three months.
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, Hopper, Falco, and Gibbons, 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, Chakraborty, and Balakrishnan, 2000). In addition, many other sensors, such as cameras 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 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 a person in locating another person.

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. A few minutes 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 needed to detect Alice for the mirror and music in the scenario to operate.

This naturalness and unobtrusiveness can pose problems. First is the issue of getting started. Once Alice knows about the mirror and how to activate it, it may be provide a natural and convenient hands-free interface. However, for the new user, the interface may pose real problems. We can expect this to be a long term problem 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 in well-known ubiquitous computing projects, like the Georgia Tech Aware Home (Kidd et al. 1999) and the Microsoft Easy Living Project (Brummit, Krumm, Kern, and Shafer, 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, Cranor, and Reagle, 1999) that many people want
to control use of personal data, especially where that data can be linked to their identity. A user model representation in this environment needs to support 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 that both she and her father would enjoy. Since the father was surprised at the change in music, and did not like it, the scenario illustrates a case where the father 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.

**Locator Scenario**

Boris 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 Alice. He indicates that he is coming to meet her. The Locator system tells Alice 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 Alice 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 is actively using his computer, meaning that he has used the keyboard or mouse within the last five minutes. The Locator system tells Alice that Boris is not available.

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

- Depending upon the situation, an application may need different levels of \textit{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 (Scott and Hazas, Addlesee et al, 2001) and ultrasound-based systems like Cricket (Priyantha, Chakraborty, and Balakrishnan, 2000).

- Similarly, different applications require different \textit{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, in this case within reasonable limits of time for Alice 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 \textit{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 describes related work on user model representations, linking it to the challenges of ubiquitous computing environments and the modelling of location and Section 7 has the final discussions, conclusions and projections of future work.

2. \textbf{Overview of the accretion-resolution representation for modelling user location}

The accretion-resolution representation has two basic operations. The first, \textit{accretion}, which involves collection of uninterpreted evidence about the user. Each piece of evidence includes the \textit{source} and the \textit{time} associated with it. The second operation is \textit{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.

Table 1 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. Each asterisk (*) in Table 1 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.** Map of sensors in lab area for current prototype. Bluetooth sensors are shown as stars 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.

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 at various times during the day. Since there are many entrances to the building, this sensor will miss him if he uses them. 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 mobile Bluetooth sensors for detecting people: range of about 10 metres; slow response to changes; false negatives as it fails to detect devices.

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 the other two. When Bob is in either of the labs, he is reliably detected by the relevant
Table 1. Example of the selected evidence available about Bob’s location. Each column corresponds to a different evidence source. \( A \) indicates machine activity sensors, \( B \) indicates Bluetooth-based sensors.

<table>
<thead>
<tr>
<th>ID</th>
<th>Time actual</th>
<th>Home</th>
<th></th>
<th>Work</th>
<th></th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>office lounge</td>
<td>foyer office</td>
<td>of®ce lab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>06:54:38</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>home</td>
</tr>
<tr>
<td>02</td>
<td>06:58:46</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>09:40:13</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>foyer</td>
</tr>
<tr>
<td>04</td>
<td>09:41:55</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>office</td>
</tr>
<tr>
<td>05</td>
<td>09:42:02</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>area</td>
</tr>
<tr>
<td>06</td>
<td>09:42:04</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>09:44:17</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>09:49:12</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>office</td>
</tr>
<tr>
<td>10</td>
<td>09:49:13</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>area</td>
</tr>
<tr>
<td>11</td>
<td>09:50:06</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12:10:04</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>seminar</td>
</tr>
<tr>
<td>13</td>
<td>13:16:27</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>room</td>
</tr>
<tr>
<td>14</td>
<td>20:10:24</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 around, over periods of several weeks. It is important to be able to easily model such changes. In addition, 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. In Table 1, these are in columns labelled The corresponding columns of Table 1 is labelled \( A \).

The last column of Table 1 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 to 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 two seconds earlier. He was also detected by each of the two nearby lab sensors in the previous 2 to 3 seconds.
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 1 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. For example, Bob might know that his machine is sometimes used by other people in the lab. He may be able to provide information about the reliability of this sensor. 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 from sensors.

The example illustrates the approach taken to location modelling using the accretion-resolution representation, in its Personis-lite implementation:

- As in Table 1, various sensors detect the user.
- Each such event is transformed into a piece of evidence indicating the evidence type (which is always an observation for a sensor), the value of the user’s location, the identity of the source sensor and the time associated with the evidence.
- This evidence is passed to Personis which accretes it, simply allowing the sequence of evidence to come in. At this point, Personis makes no attempt to interpret the evidence and it certainly does not attempt to deal with conflicting information.
- When an application asks about the user’s location (or other modelled aspects), it specifies the resolver which should interpret the evidence. As in the example of Table 1, this may need to deal with seemingly inconsistent sensor information, as in time periods 4 to 7 when Bob was detected by multiple sensors.

This example illustrates issues in modelling sensors.

- Sensors may move.
- Sensors have accuracy in terms of range (for example, for Bluetooth, about ten metres), the likelihood of false positives and of false negatives. For example, Bluetooth sensors are prone to be affected by the relative locations of Bob’s body, his phone and the detector as he moves around his office. This may cause false negatives. Some sensors, like the machine activity sensor, may give false positives.
- Maintaining a suitable history of these aspects of a sensor makes it feasible for the user to ask a personalised system about its actions in the past. This involves determining the user’s location as it would have been determined at that nominated time and reporting this along with details of how that location prediction affected the personalisation.

When the user’s location is needed, a resolver is invoked to interpret the available evidence. This needs to resolve any conflicting evidence at that time. For example, a very simple resolver might always treat the seminar room Bluetooth sensor as more
reliable than the office activity sensor for Bob. In that case, the resolver would conclude Bob was in the seminar room in time period 12.

This simple approach has considerable merit when it comes to explaining the system reasoning to the user. The explanation would simply be of the form:

> At 12:10:04, you were detected by the seminar room Bluetooth sensor and the office-computer sensor - since you cannot be in both your office and the seminar room at once, and since the seminar room is considered more reliable, the system concluded that you were in the seminar room.

Bob can easily be offered the ability to request that the system use a different resolver, for example one that treats the office-computer sensor as the more reliable. For the case of a user who often lends their Bluetooth phone but rarely allows others to use their computer, a better resolver will be different from a user who tends to do the opposite.

**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. In that work, there were three classes of evidence sources. The main source derived from logs generated by instrumentation of the text editor (Cook, Kay, Ryan, and Thomas, 1995). Essentially, these are quite similar to unobtrusive observations of people in pervasive computing environments: for example, there was a very large amount of data, it was noisy and it required interpretation.

The second main implementation of the accretion-resolution representation is the Personis user model server (Kay, Kummerfeld, and Lauder, 2002). Where the um toolkit provided a collection of C library functions, Personis has a small set of primitives for the programmer to tell evidence to Personis and to ask for the value of parts of the model. It also generalises the user model structuring. In um, components of the model were structured into a hard coded directed acyclic graph of contexts. In Personis, the context serves to define a namespace for user model components. In addition, it supports a view facility that allows the definition of an arbitrary collection of components of the model, from any contexts. This version also 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 is Personis-lite, a severely cut-down version of the Personis implementation, available as a library within an application. It has all the elements we have described above for Personis but none of the aspects that are critical for a server: security, protocols for distributed access, access authorisation mechanisms, efficient implementation for very large and complex models. 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 has been designed as a testbed for personalised ubiquitous computing environments that support scrutability of the personalisation.
Architectural overview

We now describe our approach to modelling user location by applying the accretion-resolution representation to users, devices, sensors as well as buildings and rooms. 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, Laddaga, and Naito, 2002) which call for intuitive modelling of location.

The architecture of the MyPlace system is shown in Figure 2. At the top, we show the range of sensors. At the bottom are the Personis models:

- user models for Bob and other people;
- device models for entities such as Bob’s Bluetooth enabled phone;
- sensor models for entities such as the foyer Bluetooth sensor.

The middle of the figure shows the PUB/SUB server, a Publish-Subscribe server and programs like Bob’s location modeller accept incoming data and use that to update the user, device and sensor models. We now describe how the sensor data is transmitted and managed.

![Figure 2. Overview of the MyPlace architecture.](image)

The Bluetooth sensors detect Bluetooth devices in their environment with a scan every 30 seconds. Whenever these sensors detect a device, they publish a message to the Pub/Sub server. This message includes the device’s Bluetooth ID and its own ID. Every sensor and device has its own, unique ID.

The activity sensors operate in a similar manner. They scan for mouse and keyboard activity every 30 seconds. If there is no activity, they do nothing but if there is

April 26
activity, these sensors publish a message to the Pub/Sub server on detecting activity. 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 or ...
- a timestamp;

Figure 3 shows the process involved in translating sensor data into evidence about Bob’s location when his Bluetooth phone is detected in the foyer. When it first starts, Bob’s location modeller of Figure 2 subscribes with the Pub/Sub server for messages that contain an ID matching that of all devices carried by the user or, in the case of sensors like the activity sensor, the ID for Bob. So, for example, Bob’s location modeller subscribes for all messages containing the SenseeID for his phone (the *MAC address*) and for all messages with SenseeID that is used for his system activity sensor, and with Bob’s unique ID. This has to check for network failures and re-establish this subscription as necessary.

<table>
<thead>
<tr>
<th>user <strong>bob</strong> is carrying the device with ID <strong>BobsPhone MAC address</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>location modeller for <strong>bob</strong> subscribes for messages that contain</td>
</tr>
<tr>
<td>SenseeID=<strong>BobsPhone MAC address</strong> or <strong>bob</strong></td>
</tr>
<tr>
<td><strong>Bluetooth foyer sensor</strong> detects <strong>BobsPhone MAC address</strong>,</td>
</tr>
<tr>
<td>it sends (publishes) a message containing:</td>
</tr>
<tr>
<td><strong>SourceID</strong> = sensor id for the sensor in the <strong>foyer</strong></td>
</tr>
<tr>
<td><strong>Type</strong> = sensor</td>
</tr>
<tr>
<td><strong>SenseeID</strong> = <strong>BobsPhone MAC address</strong></td>
</tr>
<tr>
<td><strong>SensorType</strong> = <strong>Bluetooth</strong></td>
</tr>
<tr>
<td>location modeller for <strong>bob</strong> receives the message and creates two</td>
</tr>
<tr>
<td>pieces of evidence:</td>
</tr>
<tr>
<td>in model <strong>foyer</strong>, component <strong>Seen</strong>, value <strong>BobsPhone MAC address</strong>, time etc</td>
</tr>
<tr>
<td>in model <strong>BobsPhone</strong> component <strong>Seenby</strong>, value <strong>foyer</strong>, time etc</td>
</tr>
</tbody>
</table>

**Figure 3.** Summary of the sequence of actions in translating sensor data into models for users, devices and sensors.

Suppose that the Bluetooth sensor in the foyer detects Bob’s phone. It does not recognise that the phone belongs to Bob. It simply records the *MAC address* for the phone and this is the SenseeID for this device. It sends its own ID ( foyer) for the SourceID, sensor as the Type and Bluetooth as the SensorType. When the Pub/Sub server receives this message, it has the subscription from Bob for messages containing the MAC address for his phone as the SenseeID. So, it sends the message on to Bob’s location modeller.
When a message arrives at Bob’s location modeller, it has to do the main work of using this piece of evidence to update the sensor and device models. In this example of the message from the foyer sensing Bob’s phone:

- the model for the sensor whose SourceID is foyer accretes a piece of evidence to reflect that it has sensed Bob’s phone. We call this component of the model Seen since it accretes evidence for each device it has seen. This evidence includes the SenseeID which is Bob’s phone MAC address combined with the location modeller program name as the evidence source. There is also the timestamp.
- a corresponding piece of evidence is stored in the device model for Bob’s phone (with its MAC address as its ID). We call this component Seenby and it holds the SourceID for the sensor and the timestamp from the message.

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.

In Figure 4, we describe the interactions between sensor, device and user models to determine Bob’s location in this type of example. At the top left, we show the user model for Bob. This includes the components:

- location to model his location,
- carrying representing the devices Bob is carrying and
- activity for the activity sensors for Bob (such as his office computer and home computer).

When an application needs to know Bob’s location, it uses the list of devices he is carrying to determine which device models it should consult to locate him. So, for example, the arrow labelled 1 links Bob’s user model to the device model for his phone. That device model has a component location for its own location and seenby which models the sensors this device has been seen by. In the figure, arrow 2 shows the case where Bob’s phone was most recently seen by the sensor located at the foyer of the building where he works.

The model for the sensor that we have called the foyer sensor also has a component for its location. The arrow into this indicates that its value was determined by the last piece of evidence of high reliability (type given), which associates a location with this sensor. In this case, the location is a value which is described as Madsen Building Foyer, specifying the building and place within it.

Note that if this 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. We have actually moved our sensors. 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.

The next step in determining Bob’s location is shown in arrow 3. The location of the sensor is resolved and a piece of evidence is placed in the location component of the device model for Bob’s phone. Its time stamp is set to be the same as the time as the most recent evidence in the foyer sensor’s seen component. This, in turn, is used to place
Figure 4. Example of the way that Bob’s location is determined by using the evidence from a sensor in the foyer which detected his phone. The figure also shows that a similar mechanism applies to other sensors.

a piece of evidence in Bob’s model. This is evidence from the sensing of phone’s location at the time that phone was last seen in that location.

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. The resolver for Bob’s location component has to reconcile the set of evidence that is available at the time period and with the time accuracy that is requested. For example, an accurate time requests his location based on evidence from the last two minutes.

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 he wants to scrutinise the way that his model has been used, he can examine this information.

Figure 4 reflects the particular case where Bob’s location is determined from sensing his phone.

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 Pub-Sub 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 Pub-Sub 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. The user models are stored in a Berkeley DB database.†

4. MyPlace interface for the invisibility scenario

In this section, we describe the interface which makes use of 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 visitor to the building and he is coming to present a seminar. John is a new graduate student.

![Two user’s view of the whole building.](image)

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 \textit{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.

The difference between the two displays is in the details of other locations that are

† http://www.sleepycat.com/
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 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.

**Figure 6.** Two user’s views of a particular room

Our building and room models are implemented using Personis-lite 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), the Nexus system (Bauer, Becker, and Rothermel, 2002) which has been integrated in the Georgia Tech Aware Home (Lehmann, Bauer, Becker, and Nicklas, 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. Experiments on modelling user locations for the Locator scenario

We now report experiments showing how the representation manages 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 (we do not deal with autonomous sensor movement in this paper);
- 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 workplace
- and six Bluetooth sensors, one at his house and the remainder in his workplace.

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 7am. Then there is a period where no evidence if available until around 9.40am 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 straight lines may equally well reflect a series of many evidence points, all from the same location, or just two points of evidence, one at the beginning and end of the lines.

![Graph showing sensor data]

**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 approximately 7am or after the last point at approximately 7.30pm.

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. As in Figure 7, this graph has one point for each piece of evidence 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 will illustrate this in the next set of experiments.

Figure 9 shows how we model when Bob is *active* at work as a basis for implementing the policy described in the third case in 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 focused on a task but us in between active use of the mouse or keyboard. The figure show 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
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.

minutes, we can only conclude that we do not know whether he is active.

Essentially, the modelling process makes use of the evidence from Bob’s activity sensor. As was shown in Figure 4, this accretes in Bob’s model in the component called activity. The graph is derived from a resolver that interprets the evidence in this component. This resolver uses only evidence coming from the activity sensor on Bob’s work computer. It ignores the evidence from the activity sensor at his home. It would also ignore evidence from any other source that contributed to this component. To create the points for Figure 9, we asked for the value of the activity component every 5 minutes. The resolver then determined the value of the user’s activity level as active if, in the previous 5 minutes, there was at least one piece of evidence from the activity sensor in Bob’s office computer. If this was not the case, the resolver returned the value unknown.

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.

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 available 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.

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 illustrates the location determined by a resolver which returns Bob’s location based on evidence from all five Bluetooth sensors at work and the one at his home. Figure 8 illustrates the values return 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 location. Bob can control the degree of spatial accuracy and which details of spatial location are available to an application that needs his location. He would do this, using a suitable interface, to select the resolver to be used by that application.

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.

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.

Flexible modelling of time accuracy of location

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. Figure 11 also illustrates the use of different time ranges on requests to the user model, the last 10 minutes in the case of the Bluetooth location sensors and 5 minutes in the case of the activity sensors.
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 reported as unavailable.

Managing inconsistent location information

We now report the extent of conflicting evidence in the above experiments. We analysed sensor evidence for the six days shown in Figure 10. In total, there were 1821 detections by any sensor, activity or Bluetooth phone detection for the user, Bob. We defined a conflict in evidence as occurring whenever the user was detected by one sensor and there was a piece of contradictory sensor evidence in the preceding 30 seconds.

There were no inconsistencies between the two activity sensors, which are several kilometres apart. There were 448 between Bluetooth sensors and all but one of these involved the nearby sensors in rooms g61a, g61b and g62 (Figure 1). There were 60 conflicts between the activity sensor in g61a and the Bluetooth sensors in the other two rooms, indicating that the user was probably working at the machine in g61a but was detected by the Bluetooth sensors in the adjoining room. Essentially, this indicates that all but one of the conflicts were associated with the overlapping Bluetooth sensors.

The one instance of truly conflicting pieces of evidence is, as shown in time period 12 in Table 1, where the user was detected at his office and in the seminar room which is too far away for him to travel in 30 seconds. It may be that the problem here is due to poor synchronisation of the clocks in the various machines in the MyPlace system. Equally, the activity sensor may have been activated by someone else. We do not know.

This analysis is limited, involving just one user over 6 days. However, it includes interesting classic overlapping Bluetooth sensors as well as two types of sensor, Bluetooth and machine-activity. On the basis of this data, it appears that complex reasoning about inconsistency may not be justified. Instead, it appears likely that determining a user’s location, availability and similar elements relevant to a ubiquitous computing environment seems amenable to a simple resolver mechanism such as those we have discussed above and which could be readily explained to a user.

One important case of inconsistency will come from cases like our overlapping Bluetooth sensors, where the technological limitations mean that it is requires other sensor evidence to determine the user’s location within the 10 metre Bluetooth range. This can be easily coded into a resolver which treats overlapping sensors as non-conflicting. This can be readily explained to a user. It also seems sufficiently straightforward, that such cases could be automatically identified even if they have not been explicitly coded into the resolvers initially.

The other class of conflict is where the evidence suggests that the user is in two places at the one time when that could not be the case and as we noted occurred once in a six day period. Our first observation is that this will only matter if the system needs to know Bob’s location at that precise point in time. Even in this case, a very simple way to deal with such conflict is to wait a few minutes until additional evidence becomes available.

The implications for user model representations is that the issues of conflicting
Performance

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 second 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.

To assess the speed of location requests, we performed repeated tests (on 880 mHz AMD-Duron) and determined that the first location request, with its associated startup costs takes 1.9 seconds. On subsequent requests the response is faster: over 1440 runs, this gave an average response time of 0.015 seconds. This indicates that an initialised system can perform within the time constraints of instantaneous, invisible interaction and even the start up time is within the time that is reasonable for the Locator scenario.

Reasoning about the user in the past

Figures 7 to 11 show information about a user over a period of time. The basic use of the accretion-resolution representation involves keeping all the evidence that has been lodged with the user model by external sources such as evidence from devices and sensors. Moreover, as explained in Section 3, the architecture of MyPlace and the management of evidence shown in Figure 4, mean that each time an application requests the user’s location, this causes a piece of evidence to be added to the location component for the user.

This means that a user can scrutinise the past actions of an application in a ubiquitous computing environment. Suppose for example, that Bob wonders about the action of an application last month. He might be prompted to do this because, for example, the application operates differently this month. We could readily build an interface that allows him to scrutinise the the value of his location as determined for the application. He can do this even if the sensor that was relevant to determining his location has been moved since last month. The model for that sensor will have the evidence that determined its location last month.

6. Related work

An excellent review (Kobsa, 2001) indicates the breadth of representations use 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 (Brajnik and Tasso, 1994) had a database of all the user models held by the system, a knowledge base of stereotypes (Rich, 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 ATMS-like approach (Doyle, 1979) and associated mechanisms. Similarly, BGP-MS (Kobsa and Pohl, 1995) represented 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 (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, McCalla, Greer, and Neufeld, 1991) where the latter two had support for truth maintenance.

These early systems dealt with the challenges of dealing with uncertainty and inconsistency with TMS-based approaches (Doyle, 1979). These are rather heavy-weight for a ubiquitous environment. THEMIS (Kono, Ikeda, and Mizoguchi, 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, 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 based 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. As Kobsa observes, the demands of personalised applications will determine the characteristics which should be provided by a user model representation.

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, 2003) for Ubi’s world, and has provided a user interface to define policies for access to the user model (Heckmann, 2003). Heckmann also has provided an interface (Heckmann, visited April 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 a critical issue for ubiquitous computing environments relates to privacy and control. For example, from the early work on active badges (Want, Hopper, Falco, and Gibbons, 1992) to the similar approach in the Cricket system (Priyantha, Chakraborty, and Balakrishnan, 2000), the researchers noted the user concerns about privacy. In that work, it was noted that the device, rather than the user was tracked. It was also observed that it is the way location data is used or logged that is often the main concern for badge wearers.

Our work is also complementary to the research into policy languages and specifications for managing privacy (Godefroid, Herbsleb, Jagadeesan, and Li, 2000) as well as middleware for policy management, such as LocServ (Myles, Friday, and Davies, 2003) and the pawS, privacy awareness system for ubiquitous computing environments (Langheinrich, 2002). These have been influenced by P3P (P3P, visited April 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, Etzioni, and Weld, 1999) as well as (Heckmann, 2003) mentioned above. 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 systems (Langheinrich, 2001). He further argues (Langheinrich, 2002) 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.

This matches well the ongoing refinements to a range of national and international privacy legislation (Kobsa, 2001). For example, the European Community Directive on Data Protection (EU, 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’.

7. Discussion and conclusions

The use of accretion and resolvers provides an extremely simple way to deal with some of the inconsistency issues which have had quite complex solutions such as truth maintenance. The whole point of modelling some components is to track changes. For
example, the user’s location should change 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. At other times, the accretion-resolution representation allows evidence to flow into the model without interpreting it. When an application needs to know the user’s location, it can request the current value of 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.

The accretion-resolution representation makes no restrictions on the nature of the resolver: it could be a very simple interpreter of the evidence list for a component or it could be any arbitrarily complex process that makes use of the evidence for several components. In practice, it is likely that simple resolvers will be adequate for many interesting situations. In our previous research systems, we had extremely simple resolvers for reasoning about user’s knowledge. Typically, resolvers take account of the reliability of each evidence source and the timestamp on the evidence. In practical applications, it is often quite effective to determine the value of a component from the most recent evidence that is of sufficient reliability.

The accretion-resolution representation is extremely simple. In this paper, we have explored whether it is still adequate for the demands of user modelling in a ubiquitous computing environment. Although the experiments we have reported are of limited scope and scale, they strongly suggest that our approach is sufficient for an interesting class of problems in such environments. It appears to have sufficient expressiveness to effectively model the user’s location and tightly related issues of availability.

It is important to emphasise the strong link between the scrutability of a user model representation and its simplicity. Intuitively, it would seem that it is easier to help users understand their user models, and the associated processes, if the representation is simple and closely linked to evidence sources that inform the user model. We want users to be able to choose the resolvers that are to be made available to different applications. It seems likely that this will be easier if the resolvers and the processes that perform are simple. We note that this support for scrutability is a critical element of user control.

A primary motivation for the design of accretion-resolution was to provide a scrutatable 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. In this paper, we have described how the design of the accretion-resolution representation supports modelling of users in ubiquitous computing environment, in a manner that should be amenable to user scrutiny. Discussion of the very important issues of the design of suitable scrutability interfaces is beyond the scope of this paper but has been explored in previous work (Apted, Kay, Lum, and Uther, 2003, Czarkowski and Kay, 2003, Uther and Kay, 2003, Kay, 1995) including some exploration of ubiquitous computing environments (Kay, Lum, and Uther, 2003).

This paper has been limited to the description of the accretion-resolution approach, in its Personis-lite implementation, as a representation of user models in ubiquitous computing environments of the type characterised in our introductory scenarios and the examples of MyPlace implementations of similar cases. We have not addressed issues of
scalability in MyPlace. For example, the current MyPlace implementation has a single central PUB/SUB server and each user’s location modeller runs on a single machine. Nor have we explored how well MyPlace could deal with a much higher rate of user model evidence, as in systems like the Bat (Addlesee et al, 2001). We have also glossed over most of the issues related to security and privacy. Indeed, MyPlace, as described here, does not even encrypt the messages.

However, 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.

In this paper we have described how the accretion-resolution representation addresses these issues. We have illustrated this in terms of the MyPlace system where the accretion-resolution representation is implemented as Personis-lite. A central contribution of this work is that we have been able to apply the accretion-resolution user model representation to users, devices, sensors and places.

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

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

**Figure 4.** 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. Horizontal lines indicate that data has been omitted at this point so that we can show those entries that are important for the discussion. The double line indicates the end of the data used for Table 1.
References


bauer02location.


Heckmann, D, “A specialised representation for ubiquitous computing and user modeling,” *Workshop on User Modelling for Ubiquitous Computing*,


