Managing private user models and shared personas

J Kay

R J Kummerfeld

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

ABSTRACT

This paper explores the question: *How should we modify and extend existing UM Languages and representations to address specific ubiquitous computing issues?* The answer we propose builds upon previous work on single user model server and exploits the user model representation to enable users to permit parts of the environment to access selective *personas*, which allow the user to provide one persona to one application and a different persona to another. We discuss the way that sensors can contribute to the user model in an ubiquitous computing environment. We also describe the ways that we address the issues of reusability and group models in ubiquitous computing.

Keywords: user model representation, scrutability, user control

1. Introduction

User modelling has an important role in ubiquitous computing. It is essential for the personalisation of user environments and it will be the repository of information that might be collected about a user from ubiquitous sensors. As ubiquitous computing is quickly becoming increasingly important, it is timely to explore the nature of user model representations for ubiquitous personalisation.

First, however, it is useful to map out the state of the art in user model representations. An excellent overview of user modelling shells (Kobsa, 2001) indicates the breadth of representations they use. He identified a trend towards lighter weight, simpler representations. Early systems tended to come from a 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 approach (Doyle, 1979). These are rather heavy-weight for an 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 will need to 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.

In this paper, we focus on our own exploration of a user model representation which supports one of the important requirements of ubiquitous computing, that of ensuring user control over the model. People have grave concerns about privacy of personal information and clearly want to have control of the way that the information is to be used (Kobsa, 2002, Miller, 2000). This is reflected in ongoing refinements to a range of national and international privacy legislation (Kobsa, 2001). For example, the European Community Directive on Data Protection (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*'.

In Section 2, we give an overview of a user model representation what we have developed for use in a range of applications. In Section 3, we describe an architecture for personalisation in ubiquitous applications and then, in Section 4, we describe how our representation supports the needs of that architecture. Section 5 discusses the reuse of user models and the role of group models, followed by the final discussion and conclusions.

2. Overview of the accretion representation

The accretion representation and architecture (Kay, 1995, 2000, Kay, Kummerfeld, and Lauder, 2002) is an extremely simple but flexible approach to modelling users. It consists of *components* each of which models an arbitrary aspect of the user. It distinguishes knowledge, beliefs, preferences and other attributes of the user, including their personal attributes like names, height, date of birth and other arbitrary aspects like their location. Each of these has a type, such as boolean, string, number. The representation makes no built-in assumptions for any of these types.

We illustrate an example in Figure 1, which depicts flow of information about a component which models the user's location. There are

three sensors contributing information about this component. The first column is a Bluetooth-based system which sends a piece of evidence when it detects the user in their office. The second is a similar sensor for the tearoom. Both of these sensors send positive evidence in each time period that they detect the user. Sensor 3 sends a piece of evidence about the user in each time period, a + when there is activity at the user's office computer and a − otherwise.

Time	Sensor 1 Office	Sensor 2 Tearoom	Sensor 3 Computer
1	+		−
2	+		−
3	+		+
4	+	+	−
5		+	−
6		+	−
7		+	+
8		+	−
9	+	+	−
10	+		−

Figure 1. Example of evidence lists in time series for a component modelling user location. Sensors 1 and 2 are Bluetooth-based sensors. Sensor 3 tracks activity on the user's office computer.

The name of the representation follows from its model for managing incoming information about the user. Essentially, each piece of evidence about a component is simply added to a list. So, over time, it *accretes*. As in Figure 1, as various sensors detect the user, each can volunteer evidence. The core of the accretion representation is this simple process of collecting this growing list of information.

The sensor information may appear inconsistent, as in time periods 4 and 9 where both the tearoom and office sensors' evidence reports the user's location in these rooms. This is likely to happen when the user is between the two sensors and is detected by both, as they move between them. Another example of possible inconsistency is in time period 7 where one sensor reports activity at the user's office computer while the other two sensors appear to indicate the user is in the tearoom.

Sources are only permitted to contribute to the model if they are authorised to do so. In an ubiquitous computing context, a user might allow sensors in their office to contribute to their location model but the user might not allow sensors in their home to do so. Another user might want different restrictions on the sensors allowed to contribute to their model.

When the value of a component is needed, a *resolver* is invoked. The 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. 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.

Following this approach, at time period 7 of Figure 1, we have noted that the evidence + from the tearoom sensor contradicts that from the office computer evidence source. One suggests the user in the tearoom while the other suggest that the user is active at their terminal in their office. There are many possible explanations for this conflict, such as: someone else might be using the computer and Sensor 3 may simply be detecting another person's activity at the user's machine; Sensor 3 may be in error for some other reason like a book falling on the keyboard and keeping a key depressed; the user may have lent their mobile phone to someone else and that person may be in the tearoom being detected; Sensor 2 may be oversensitive and often detects the user when they are actually in their office. This sort of situation will be typical of those faced in the reasoning about users in a ubiquitous computing environment.

A simple resolver might always treat Sensor 2 as more reliable than Sensor 3. In that case, the resolver would conclude the user was in the tearoom at time period 7. Note that this simple approach has considerable merit when it comes to explaining the system reasoning to the user. The explanation would simply be of the following form:

At time period 7, you were detected by the tearoom sensor and the office-computer sensor - since you cannot be in both your office and the tearoom at once, and since the tearoom sensor is generally considered more reliable, the system concluded that you were in the tearoom.

The user could easily be offered the ability to request that the system use a different resolver, for example one that regards Sensor 3 as more reliable. For the case of a user who often lends their Bluetooth phone but rarely allows others to use their computer, the right resolver will be different from a user who tends to do the opposite.

Note that this 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. If, as is likely in ubiquitous computing contexts, most of the time, no application needs to know the user's location. At those times, we simply allow 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.

We have three main implementations of the accretion representation. First, it was built as the um toolkit (Kay, 1995) where it was used 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 editor (Cook, Kay, Ryan, and Thomas, 1995). We never put this directly into the user model. Instead we built a collection of analysis tools which used that data to infer what the user appeared to know or not know. Some of these were very simple: the fact that a user typed a complex regular expression command that produced no errors or warnings was interpreted as direct evidence that the user knew that command. Some were more complex. In particular, the mouse commands could easily be selected accidentally so they were only interpreted as correctly used when there were several aspects of the context which were consistent with their correct use. For example, one command creates multiple windows on a single file: the analyser only gave a piece of evidence indicating correct use if the file was large enough to need two windows to display different parts, if there was additional activity afterwards and if the user was already past the novice stage. Each different analysis tool was treated as a separate evidence source. The second form of evidence came from the automated tutor which added evidence to the model when it sent advice to the student. The third form of evidence came from the user's interaction with the user model viewer tools: these enabled the user to state whether they knew aspects or not and any such information generated evidence which was added to the model. We built several resolvers, all simple. Notably, the one used by the automated tutor treated evidence derived from the logs as more reliable than that from the user.

The second main implementation of the accretion representation is in the Personis (Kay, Kummerfeld, and Lauder, 2002) user model server. Where 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.

The third implementation is Personis-lite, a severely cut-down version of the Personis implementation, available as a library for use 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.

The example in Figure 1 highlights the serious efficiency burden of this approach as described to this point. This is clearly a serious problem if we want the user to be able to scrutinise their user model. Clearly, we could not allow the evidence lists in the model to continue to grow without restriction. We need some *destruction* operations. These are explicit processes which operate on parts of the model. In particular, we have a *use-by* date on evidence. So, for example, evidence from a location sensor may have a short use-by date so that it is destroyed after a short time. On the other hand, it might be allowed a longer use-by date so that patterns in the user's behaviour can be identified over time, such as was done in Doppelganger (Orwant, 1995) with a somewhat similar representation to accretion.

A primary motivation for the design of the *accretion* was to provide a *scrutable* representation, where the user could examine the user model, the *value* of its components 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.

3. Architecture of distributed, ubiquitous user models

We now describe how we envisage the accretion representation operating in an ubiquitous computing environment. Figure 2 outlines an architecture of the personalisation in a such an environment. This builds on our previous architecture (Kay and Kummerfeld, 1994).

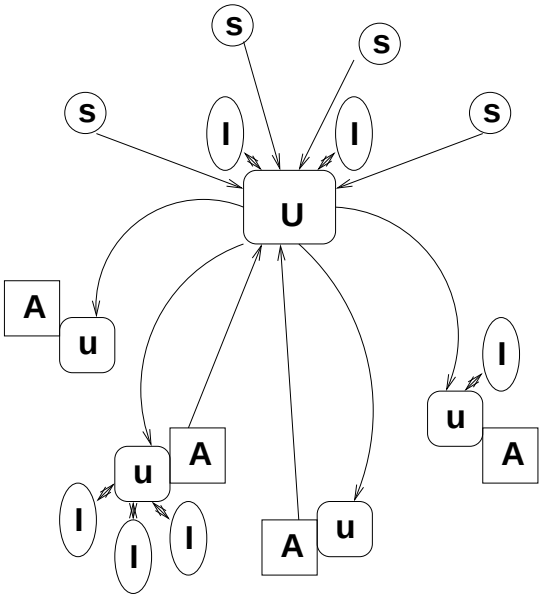

Figure 2. Architecture for distributed personalisation. U is the single user model for a person. Each application, A, has its own partial user model u. The central model and partial models may have local inference sources, I. Each s represents sensor data coming to the user model.

There is one, definitive user model, U, for an individual, shown at the centre top. It could be implemented using the Personis server (Kay, Kummerfeld, and Lauder, 2002) with strong security and authentication mechanisms. It is held at a physical location decided by the user and would be controlled by the user.

There are several partial models, or *personas*, shown as u. Each is intended for one application, shown as A. The user controls which aspects of U are available in each persona. For example, a meeting organiser application may be restricted to a persona with the user's preferences for modes of communication and their timetable, but it would not include any other user model information. The arcs indicate that applications are authorised to access a part of the user model, U, which they would hold in their local u.

We envisage that the central server, U, would need to be supported by a system like Personis while the local partial user models may either be full Personis servers also or just the cut-down Personis-lite. In the context of a ubiquitous computing environment, there may be many low powered computational elements within the environment. These may be able to exploit light and small personas for users in that environment. These elements would be able to incorporate the Personis-lite support for accretion-based reasoning at modest cost in memory and speed.

The inferences sources, **I**, are the internal reasoning mechanisms and each of these is a potential supplier of evidence. These operate by using resolvers to access the values of components in the user model and from this information, they make inferences about the user and these are stored as evidence in the user model. The inference mechanisms need only be invoked as needed. For example, they may be triggered at a set time each day or by an event. This form of evidence also accretes in the model. A deletion operation on this evidence has a quite different effect from a deletion of evidence from external evidence sources such as sensors. The critical difference is that, as long as all external evidence is kept, the inference sources can be run so as to simulate the way that they would have operated at any arbitrary time in the past.

One important form of inference is the stereotype. Another is based on knowledge of the domain. As shown in the figure, inference sources may reside at the user model **U** or with *personas*. Users control which inference sources may be used for the user model and the personas. For example, a user may allow just one application to use the set of inferences associated with the *new employee* stereotype.

The figure also shows sensors, **s**, which can also contribute user modelling information to the user model. We do not show these in association with applications since they, like sensors, can contribute evidence to the user model. Although we do not show it in this figure, it may be that these sensors might be able to improve their

performance if they held a small amount of user model information. Our comments on the use of the Personis-lite implementation for the user model personas **u** apply in this case.

The content of each persona is controlled by the user. There are several levels of such access control in the accretion representation:

- evidence sources - so that the user can allow an application access to values based on just the evidence sources allowed to that application's persona. For example, in Figure 1, one persona may have access only to the value of the component, based upon evidence from Sensor 1. Any application using this persona would be blind to all other evidence in the user model.
- component - it is possible to control access to each individual component, allowing it to be included in a persona, or not;
- view - the main mechanism for defining the collection of components in a persona;
- context - our structuring mechanisms for a collection of user model components and associated structures, such as a movie context holding components about the user's movie preferences. The context defines a namespace and it holds definitions of components, default access modes across the context and it also holds the view definitions.

For the purpose of managing personas, the main mechanism is the view.

4. Scrutability

Users need to be able to scrutinise and control their user models. This has been foremost in the design of Personis, from its very beginnings (Kay, 1990) through to the most recent work (Kay, Kummerfeld, and Lauder, 2002). We now discuss this aspect of the accretion representation in the ubiquitous computing context.

The core of the representation is the simple model that evidence accretes over time. This is important for scrutability on two levels. Firstly, if a user is to be able to scrutinise and really develop an understanding of the way that their user model operates, we need to strive for *simplicity*. Some complexity is unavoidable because of the nature of the user modelling enterprise: we have to accept the complexity due to the number and range of sources of user model evidence and the varied ways that the model might be employed in an ubiquitous computing environment. However, the representation should manage these as simply as possible with a collection of simple mechanisms.

The second critical aspect of pure accretion is that it ensures that the systems has all the evidence that may have affected any personalised action. The user can scrutinise the full list of evidence about each component of the model.

Moreover, that evidence is in a form suited to scrutability. Each piece of evidence is kept with details about the evidence source. Each evidence source that is authorised to contribute to the user model has descriptions for the way that the evidence source operates so that a user scrutinising the model can access these. The other parts of the evidence, its type, time-stamp and the value it supports, can be also be explained to the user. The user model allows an evidence source to contribute only to the parts of the model where they are authorised to do so. This means that the user can control which sources may contribute to which parts of their model. This means, for example, that the user can authorise evidence from a presence-sensor in their office but *not* the one in the tearoom.

Where the systems mentioned earlier had quite sophisticated mechanisms for dealing with uncertainty and inconsistency, the accretion representation takes a very simple but quite flexible approach. Firstly, the accretion of evidence is independent of its interpretation. This means that if conflicting evidence comes into the model over a period of time, there is no attempt to reconcile this until it is actually needed. At that point, the application which needs the values of parts of the user model accesses these via *resolvers*. The goal of simplicity associated with scrutability means that these should be as simple as possible. In previous work, modelling student knowledge and movie preferences, very simple resolvers gave good results, which could be explained quite simply. It is desirable to strive for similar resolvers in ubiquitous computing tasks.

The deletion of evidence has important implications for scrutability. On the one hand, it will be harder for users to scrutinise and understand their user model if it is allowed to grow with large amounts of evidence that is too old or insignificant to affect the value of the components of the model. On the other hand, keeping complete sets of the evidence from external sensors means that the user can scrutinise past behaviour of a personalised system.

To this point, we have listed many the aspects which can be controlled by the user: the authorisation of processes to create new parts of the model, authorisation of evidence sources and inference mechanisms to reason about the user, authorisation of resolvers for use by applications; authorisation of applications to access the evidence produced by each evidence source; authorisation of parts of the model to be available to applications. In spite of the simplicity of each part of the accretion architecture and representation, this still makes for considerable complexity for the user to cope with. Enabling this poses a signifi cant user interface challenge. People may just opt out, unless the user interface is adequate.

The notion of scrutability might appear to be at odds with the core goals of invisibility (Weiser, 1994) in ubiquitous computing research. IEEE Pervasive Computing holds up the goal of building ubiquitous systems that are 'gracefully integrated with human users'.† Our approach presumes that most users will want invisible personalisation most of the time. However, when the user wants to know why systems are performing as they are or what the user model believes about them, they should be able to scrutinise the model and the associated personalisation processes. For example, most mobile phone users do not want to have to know the details of the current cell supporting the phone. However, phones can provide this information readily and users can and do choose to check it.

In general, an established system should operate without bothering the user. However, suppose that the user happens to receive a message via a phone which rings for them as they walk into their colleague's offi ce. If the user wants to know why this happened, they should be able to access the parts of the model responsible for tracking their location and delve into the details to see which evidence source provided the relevant information at that time. At this point, the user may wish to alter any access modes which effected personalisation actions that the user did not want. It may be possible to defi ne quite simple control mechanisms for ubiquitous applications: a simple user action for disabling an application from within the ubiquitous computing environment may be suffi cient, with good support for the user to later use conventional systems to refi ne the details of their personas.

This view of scrutability of personalisation in ubiquitous computing is similar to the notion described as seamfulness (MacColl, Chalmers, Rogers, and Smith, 2002) which argues for 'overt, robust, flexible, simple and manipulable' systems. Our representation approach is in the spirit of recombinant computing (Newman et al, 2002) and the persona which can be viewed and scrutinised by the user is a similar notion to the calendar mirror. The need for user control of personal information has been widely acknowledged, for example (Boyd, Jensen, Lederer, and Nguyen, 2002) and (Abowd and Mynatt, 2000) who observe that 'One fear of users is the lack of knowledge of what some computing system is doing, or that something is being done ''behind their backs'' ' (p51) and that 'The next step is to allow those being sensed to have control to either stop this activity or at least control the distribution and use of the information.' (p52)

† http://www.computer.org/pervasive/faq.html

5. Reusability and Group models

To this point, we have focussed on the representation of individual user models. We have seen that there is an important role for reuse of parts of the model across different applications. Where these are disparate, as in the case of recommenders, teaching systems, personalised media delivery systems and the like, this essentially means that the user does not need to train each. Instead, once a relevant collection of components has been defi ned and values for them established, the user can make them available to new applications. This reduces the *startup* problem for personalisation, where the application needs to build up a user model before it can perform useful customisation.

In ubiquitous computing, it will be critical to reuse user models at a different level. If we can build a statistically valid *group* or *stereotypic* model (Rich, 1989), a user who provides minimal information about themself can be provided with an initial user model which is based on data from a group of similar people. Just this approach applies in the most common recommender systems such as several at Amazon.com. In these, as in the case of ubiquitous computing, we may have a single point of information about the user, perhaps a request for information about a single book in the case of Amazon, perhaps a small trace of user behaviour in the ubiquitous computing case.

One of the earliest exploration of user modelling for ubiquitous computing was Doppelganger (Orwant, 1995) which used active badges and smart chairs to track user activity within a

building and on computer systems and phones. It had a very interesting, elegant and simple group notion, the *community*. It differs from stereotypes in that a male always contributes data to the group model for males. However, predictions about the user are based on a weighted average of information from the combination of communities that they best match. The simplicity of this notion should be very appealing for ubiquitous computing. It has not been explored since Doppelganger.

One of the challenges of building group models is that we need quality data about people. So, for example, if we would like to model the common patterns of people's movement around a building, we need data about that. Why should a user contribute their personal data? One reason would be that there is personal benefi t. For example, if a user were allowed access to data about the location of other people, only if they were willing to allow the system access to their own data, this appears to be fair and may enable data to be aggregated. A similar approach has been applied in the Educo system (Kurhila, Miettinen, Nokelainen, and Tirri, 2002) where people reading documents from an online selection were allowed to see who else was reading each document. This could serve as a basis for establishing ad-hoc online discussions about a document. Users were only allowed to see information about others location in the virtual document space if they allowed themselves to be visible.

6. Discussion and Conclusions

We have described aspects of the accretion representation for scrutable management of user models. It has not been deployed in an ubiquitous computing context: its main use was in the modelling of knowledge of a text editor where the bulk of the user model information came from a collection of evidence sources, each based on analysis of logs from monitoring users of the text editor. This monitoring ran for 10 years and user models were built from data collected over two years. In that context, it was evaluated in terms of its capacity to support tutoring. Its scrutability was also evaluated both in a small qualitative experiment and in a large fi eld study (Kay, 1995). The latter indicated that some users did scrutinise their model extensively over the 8 weeks of the study.

Ubiquitous computing will offer quite different challenges and the architecture we have described envisages a broader ranging user model. It is yet to be determined whether the simple accretion, resolution and, where necessary, deletion of evidence can support the reasoning needs of ubiquitous systems. For example, if we want to model users location transitions, a natural representation would be a Markov Model. Whilst accretion maintains evidence in a form suitable for learning such a model for the user, this would need to be managed by one of the inference sources shown in our architecture. When we need a prediction for the user's future location, that Markovian inference source could be triggered to supply evidence with a prediction of the location in the future; a suitable resolver would be able to use that to drive the system. This approach has yet to be explored.

The various forms of user control facilitate the creation of personas, each intended for use by particular applications. We suppose that a new application's persona would be created by a process where the application proposed the components and evidence sources to be included in the model. The user would examine these, selecting those components and evidence sources they were happy to allow into that persona, and restricting others. Good interfaces for this task are an important area for future work. One promising approach for some classes of user model is a visualisation of aspects of the information to be allowed into the persona, as in (Uther, 2001, Apted, Kay, and Lum, 2003) and a very similar approach to multiple personae (Boyd, 2002) but called facets of identity.

One might argue that one of the goals of ubiquitous computing is that the user should not need to be aware of all the computation embedded in the environment. The computational elements should *disappear*. Indeed, we agree that most of the time, the user should not need to be aware of the details of their user model and the way that the personalisation operates. However, the user model is essentially personal information. This means that users have a right to have access to it. Users also have the right to control the personalisation that machines effect on their behalf. There will be times that the user will become aware of the personalisation: in particular, if something unexpected happens or the system gets things wrong, the user will become aware. The user may even become aware because the system does something that they like very much. Or they just might become curious about how the personalisation work because they happen to be curious people. At that point, we need to be able to support

the user in tracking down just what is happening and why. The ability to scrutinise the user model and its interpretation of the evidence available to it will be critical at that point.

The stereotypes which can be used for default reasoning about users also have some promise in this area. It may be possible to build several stereotypic sets of personas that people might use as a starting point for their own persona. This could be supported by a visualisation such as (Uther, 2001). There may also be possibilities to stereotype some of the classes of personae: for example, the user may create a restricted *advertiser* persona and this would be the default for a range of advertiser applications. So, for example, in the future ubiquitous computing environment, the user who walks into the supermarket might be presented with personalised advertising, based on the persona they have made available for such applications. This might include information that might be quite general or, alternatively, it might have the detailed set of their current shopping needs, something they might regard as soon-to-be-available to the environment (after they had passed the checkouts).

We have given an overview for a representation and architecture for user modelling in ubiquitous computing. Much of the detail of both have been omitted. There are also many problems that we have not mentioned. One important class of these relate to the way that the user model interacts with the rest of the environment. In particular, there are many issues to address in the way that sensor data makes its way to the user model. There are open questions about the ownership of the sensor data: does this belong to the user or the object that triggers the sensor event?

Our current work has been in the areas of both context-aware systems and user modelling systems. We are developing an architecture for a general, large scale context management system that uses a distributed repository for context data. We have built an early prototype of this architecture with Bluetooth access points to provide location information and simple applications to track people with Bluetooth phones. Our Personis user model server is being used in several other applications and we plan to integrate it with the context repository to provide personalisation services to context aware systems.

References

Abowd and Mynatt, 2000.
Abowd, G and E Mynatt, "Charting past, present, and future research in ubiquitous computing," *ACM Transactions of Computer-Human Interaction,,* 7, 1,, pp. 29-58. (2000).

Apted, Kay, and Lum, 2003.
Apted, T, J Kay, and A Lum, "Visualisation of learning ontologies" in *Proceedings of AIED Conference, 11th International Conference on Artifi cial Intelligence in Education,* p. to appear, IOS Press (2003).

Boyd, 2002.
Boyd, D, "Faceted ID/entity: managing representation in a digital world," *Naster of Science in Media Arts and Sciences,* 2002, p. accessed http://smg.media.mit.edu/papers/ March 2002, MIT (2002).

Boyd, Jensen, Lederer, and Nguyen, 2002.
Boyd, D, C Jensen, S Lederer, and D Nguyen, "Privacy in digital environments: Empowering users.," *Workshop abstract in Extended Abstracts of the ACM Conference on Computer Supported Cooperative Work (CSCW 2002).,* New Orleans, Louisiana. (2002).

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

Chin, 1989.
Chin, D, "KNOME: modeling what the user knows in UC" in *User models in dialog systems,* ed. A Kobsa and W Wahlster, pp. 74-107, Springer-Verlag, Berlin (1989).

Cook, Kay, Ryan, and Thomas, 1995.
Cook, R, J Kay, G Ryan, and R C Thomas, "A toolkit for appraising the long term usability of a text editor," *Software Quality Journal,* 4, pp. 131-154, Chapman and Hall, London (1995).

Doyle, 1979.
Doyle, J, "A truth maintenance system," *Artifi cial intelligence,* 12, 3, pp. 231-171 (1979).

Huang, McCalla, Greer, and Neufeld, 1991.
Huang, X, G I McCalla, J E Greer, and E

Neufeld, "Revising deductive knowledge and stereotypical knowledge ina student model," *User Modeling and User-Adapted Interaction,* 1, 1, pp. 87-116 (1991).

Kass, 1991.
Kass, R, "Building a user model implicitly from a cooperative advisory dialog," *User Modeling and User-Adapted Interaction,* 1, 3, pp. 203-258 (1991).

Kay, 1995.
Kay, J, "The um toolkit for cooperative user modelling," *User Modeling and User-Adapted Interaction,* 4, 3, pp. 149-196, Kluwer (1995).

Kay, 2000.
Kay, J, "Accretion representation for scrutable student modelling" in *ITS'2000, Intelligent Tutoring Systems,* ed. G Gauthier, C Frasson, and K VanLehn, pp. 514--523 (2000).

Kay, 1990.
Kay, J, "um: a user modelling toolkit," *Second International User Modelling Workshop,* p. 11, Hawaii (1990).

Kay and Kummerfeld, 1994.
Kay, J and R J Kummerfeld, "Customization and delivery of multimedia information" in *Proceedings of Multicomm 94, Vancouver,* pp. 141-150 (1994).

Kay, Kummerfeld, and Lauder, 2002.
Kay, J, R J Kummerfeld, and P Lauder, "Personis: a server for user models," *Proceedings of AH'2002, Adaptive Hypertext 2002,* pp. 203 -- 212, Springer (2002).

Kobsa, 2001.
Kobsa, A, "Generic User Modeling Systems," *User Modeling and User-Adapted Interaction - Ten Year Anniversary Issue,* 11, 1-2, pp. 49-63 (2001).

Kobsa, 2002.
Kobsa, A, "Personalized Hypermedia and International Privacy," *Communications of the ACM,* 45, 5, pp. 64-67 (2002).

Kobsa, 2001.
Kobsa, A, "Invited Keynote: Tailoring privacy to user's needs" in *Proceedings of the User Modeling Conference,* pp. 303 - 313, Springer (2001).

Kobsa and Pohl, 1995.
Kobsa, A and W Pohl, "The user modeling shell system BGP-MS," *User Modeling and User-Adapted Interaction,* 4, 2, pp. 59-106, Kluwer (1995).

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

Kurhila, Miettinen, Nokelainen, and Tirri, 2002.
Kurhila, J, M Miettinen, P Nokelainen, and H Tirri, "Dynamic profiling in a real-time collaborative learning environment," *ITS 2002m LNCS 2363,* pp. 239 - 248 (2002).

MacColl, Chalmers, Rogers, and Smith, 2002.
MacColl, I, M Chalmers, Y Rogers, and H Smith, "Seamful ubiquity: Beyond seamless integration," *Proceedings of Ubicomp 2002 Workshop on Models and Concepts for Ubiquitous Computing,* Gothenburg (2002).

Miller, 2000.
Miller, C, "Report SS-00-01," *Working notes of the Adaptive User interfaces AAAI Spring Symposium,* pp. 80--81, Stanford, CA (2000).

Newman et al, 2002.
Newman, M, S Sendivy, C Neuwirth, K Edwards, J Hong, S Izadi, K Marcelo, and T Smith, "Designing for serendity: supporting end-user configuration of ubiquitous computing environments," *Designing Interactive Systems* (2002).

Orwant, 1995.
Orwant, J, "Heterogenous learning in the Doppelganger user modeling system," *User Modeling and User-Adapted Interaction,* 4, 2, pp. 59-106, Kluwer (1995).

Paiva and Self, 1995.
Paiva, A and J Self, "TAGUS - a user and learner modeling workbench," *User Modeling and User-Adapted Interaction,* 4, 3, pp. 197-228, Kluwer (1995).

Rich, 1989.
Rich, E, "Stereotypes and user modeling" in *User models in dialog systems,* ed. A Kobsa and W Wahlster, pp. 35-51, Springer-Verlag, Berlin (1989).

Union, 1995.
Union, The European Parliament and the Council of the European, *European Community Directive on Data Protection,* http://www.doc.gov/ecommerce/eudir.htm

(1995).

Uther, 2001.
Uther, J, "On the visualisation of large user models in web based systems," PhD Thesis, Department of Computer Science, University of Sydney (2001).

Weiser, 1994.
Weiser, M, "The world is not a desktop," *Interactions,* 1, 1, pp. 7-8 (1994).

Supporting Proactive 'Intelligent' Behaviour: the Problem of Uncertainty

Hee Eon Byun and Keith Cheverst

Distributed Multimedia Research Group,
Department of Computing,
Lancaster University
Lancaster, LA1 4YR
e-mail: {byunh, kc}@comp.lancs.ac.uk

ABSTRACT

This paper describes our exploration into some of the inherent uncertainty issues that arise when designing for proactive and 'intelligent' behaviour within ubiquitous computing environments. We describe both previous and current work that has motivated our current examination of uncertainty issues and also discuss some possible implications for user interaction that arise when providing proactive behaviour based on rules induced from (potentially uncertain) context history.

1. Introduction

Over the past decade, one of the predominant trends in computing has been "*ubiquitous computing*", a term that was first proposed by Weiser in the early 1990s. His vision was of increasing the productivity or welfare of a user situated in a computer-everywhere environment by supporting human assistance in an intimate way [17]. One research domain that requires the computer-everywhere model of ubiquitous computing is that of the "*intelligent environment*" [6]. In this domain, a wide range of physical devices (e.g., lights, audio/video equipment, heaters, air conditioning equipment, windows, curtains, etc.) can be controlled automatically based on the context of a house/office and the preferences of inhabitants (e.g., preferred temperature, preferred light level, energy consumption policies, etc.). One promising technique for achieving such intelligent environments is "*context-aware computing*". The term 'context-aware' has been defined as "systems [that] adapt according to the location of user, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time" [7].

In our previous works [2] and [3], we have suggested that the history of contexts could be extremely valuable in enabling the current levels of context interpretation (context aggregation [7], context synthesis [15], and context fusion [4]) to be enhanced. In more detail, by noticing patterns from the user's context history (including the user's behaviour) a properly designed system may start to exhibit appropriate "intelligent" or more specifically "proactive" behaviour. For example, by identifying recurring patterns in a user's context history (based on contexts such as user location, calendar information etc.) it may be possible to utilise machine learning techniques in order to determine that a certain user has a regular meeting schedule. An intelligent reminder system could then take the proactive step of reminding the user of her meeting.

The traditional method for providing proactive behaviour is to use predefined rules [3]. One limitation of predefined rule-based adaptation is that the user must reconfigure the system in order to reflect changes to her routine. Performing such reconfigurations could be a frustrating or annoying task for the user. Therefore, we advocate "modelling-based proactive adaptation. In more detail, by observing the routine of the user in an intelligent environment there is the potential to infer appropriate rules from accumulated context history in order to learn rules and subsequently provide dynamic adaptations.

Uncertainty is an important consideration when supporting either type of proactive adaptation,. In more detail, various sources of uncertainty play a part in this approach, namely: uncertainty from sensing context (e.g. margins of error in a given sensor), uncertainty that arises through interpreting contexts (e.g. deriving higher level contexts from a group of lower-level or raw contexts). When supporting but it is proactive modelling-based adaptation uncertainty also arises when inferring rules from context history (fully described in section 3).

One potential problem that may occur with intelligent proactive systems is that the system may behave in ways that do not fit well with the user's mental model of the system. Although we cannot avoid the potential situation of the system acting in an unexpected way, our proposed solution is to provide users with explicit and meaningful explanations when such unexpected behaviour occurs [1]. Providing such an explanation to the user may in turn enable the user to provide some feedback to the system in order to enable the system to refine the rules which it had inferred from the context history. Figure 1

illustrates the way in which our approach involves the user in accepting proactive behaviours and potentially causing the modification of the system's adaptation rules. Crucially we believe that keeping the user involved in the loop may also involve informing the user of possible implications of inferences based on context history that may contain uncertainties.

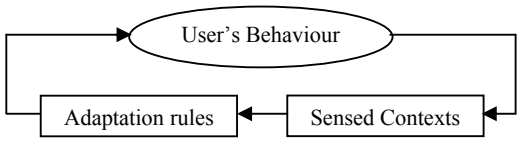

Figure 1. The relations between a user, the user's behaviour, and contexts.

The structure of the remainder of this paper is as follows. In section 2, we describe some of our previous work that has played a key role in motivating our exploration of the uncertainty issue. In section 3, we investigate some of the potential sources of uncertainty and consider some possible countermeasures. Interaction issues between a proactive system and a user are discussed in section 4. Finally, a summary of our investigation on the issue of uncertainty is presented in section 5.

2. Previous Work

2.1 PDS scenarios

An early motivating scenario for our research into proactive context-aware behaviour was the Personal Digital Secretary (PDS) [2]. This application idea emerged from considering how the classical remembrance agents, for example, Forget-me-not [12] and CyberMinder [8], might be extended using the techniques of user modelling and machine learning in order to support a user's daily activities in an everyday computing setting [1]. The PDS was designed based on the assumption that it would support a user's daily activities beyond the role of a remembrance agent or reminder. For the conceptual structure of our PDS and detailed explanations, please refer to [2].

The PDS scenario proved extremely useful for helping us to explore how the paradigm of context-aware computing could be usefully augmented by the utilisation of user modelling and machine learning techniques. Some examples of typical scenarios that reveal proactive usages of the PDS are described below.

- *Scenario A:* When a user passes by a theatre, the PDS can notify the user that the theatre is playing one of the user's favourite movies. This type of functionality could be realised by a context-aware system (such as GUIDE [5]) by utilising both the location context and information about the user's preferences (such preferences could have been learnt, pre-defined or a combination of both).
- *Scenario B:* If a user is in an intelligent environment and the user commands some action, for example, 'close curtains in the living room', the PDS could modify its user model and, over time, learn that an appropriate context-aware behaviour is to close the curtains when it gets dark outside.
- *Scenario C:* If a user participates in a meeting at 10 am every forth Monday, the PDS might learn the pattern of this regular meeting and remind the user to prepare for the meeting at an appropriate time. However, a more sophisticated level of learning would be desirable in order to enable the system to realise when such a notification is inappropriate, for example, when the user is on holiday.
- *Scenario D*: if a user makes a rule to hold the room key when leaving his/her office after 6pm, this can be captured in a user model. Consequently, when the user is about to leave his/her office without the room key after 6pm, the PDS could warn the user before he/she gets locked out of the office!

2.2 Calculating the Level of Security Risk in a User's Office

We first examined the potential of context-history together with user modelling and machine learning techniques) by considering a specific scenario for calculating the level of security risk in a user's office [3]. In this work, we explored a number of different approaches including the use of a Naïve Bayes Classifier, which was used to calculate the conditional probability of the user being in her office. In more detail, a set of rules were predefined for deciding the level of security risk in a user's office and a simple context history was artificially built in order to detect exceptional cases against the predefined rules: the first rule: a high security arises if the door is open when the user has left the office during her office hours, and the second rule: a low security risk arises if the door is closed (but not locked) when the user has left the office during her office hours.

The aim of the work was to examine two questions: (1) could patterns of the user's behaviours be properly extracted from the context history? (for example, the user's likely returning to her office given that a cup in the office contains hot coffee) and (2) could the extracted patterns be used for appropriate decision making? In this theoretical test case, we found that context history did have a strong potential for supporting dynamic adaptations in a ubiquitous computing environment even though there were a number of issues to be challenged [3]. The next step of our research (and the experiment described in this paper) was to build a prototype system that could support proactive modelling-based adaptations in a user's office.

2.3 Context-Based Intelligent Environment Control

In order to ascertain the feasibility of supporting proactive modelling-based adaptations in a user's office our current work has involved the design and implementation of a system to:

i). Utilise context history in order to learn the patterns of the user's behaviour in a physical office environment; and

ii). Support proactive modelling-based adaptations (opening/closing the window, turning on/off the fan), based on both the patterns learned (represented as a generalised set of rules) and the state of the physical office environment (realised through a set of sensors).

Contexts considered in the experiment are temperature, humidity, noise level, light level, the user's task (keyboard typing or not), the status of window, the status of fan and the status of blind. Actuators are needed to open/close the window, turn on/off the fan, and open/draw the blind in the user's office. Our system collects and accumulates the contexts as a context history. Next, it induces a set of rules from the context history. Currently, the rules represent the user's preferences to the status of window (i.e. this is currently the only target function considered) Based on these rules, our system can provide a suggestion to the user when the physical environment in the office changes, e.g. if the temperature rises above a given threshold value. It is important to note that whenever the system suggests an adaptation, e.g. "shall I open the window?" the user can dismiss the suggestion.

Our system comprises two databases (one for storing context history, the other to store the user model) and three main modules (a context manager, an inference engine, and an adaptation manager) as illustrated in figure 2. The context manager collects context from sensors in a clock-based fashion and encodes numeric context values into symbolic representations. If there is at least one change in a symbolic context value (e.g., if the temperature changes from "20°C ≤ mild ≤ 24°C" to "hot > 24°C", or if the user records an appropriate action, such as opening the window), then the context manager generates a context-changed event and stores the context in the context history. The adaptation manager listens for the context-changed events being raised by the context manager. Next, the adaptation manager decides which adaptation must be made based on both the current situation (e.g. current temperature) and the user model (which contains a set of learnt or predefined rules that represent the user's preferences). Learnt rules are placed in the user model by the inference engine which is responsible for extracting rules based on the context history. These rules can be learned through either on-line or off-line processing. At this stage of our research, an off-line learning method is adopted.

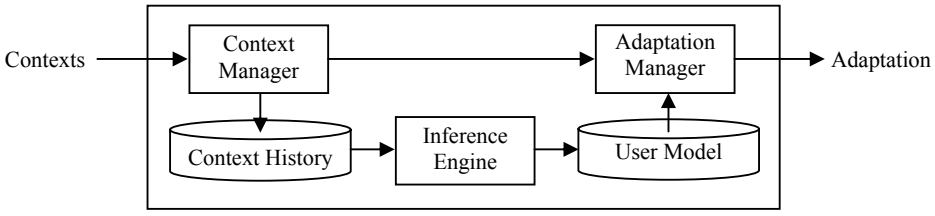

Figure 2. The design for providing dynamic adaptations.

As an initial result of our experiment, we found that there was a phase of transition for the initiation of adaptations. In more detail, during the first day of our experiment (t0 in figure 3) our system could not learn any rules because there was no context history. At this stage, a set of predefined rules was used for providing adaptations, for example, if the temperature is higher than 24°C then open the window. From the next day, our system started to learn the user's preferences from context history. However, it took about two weeks (from t0 to t1 in figure 3) to accumulate an appropriate size of context history for properly learning the user's preferences. Our system gradually learned the user's preferences and provided more refined suggestions as time went by.

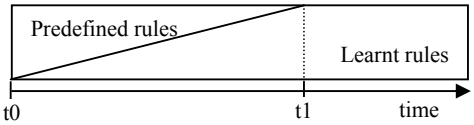

Figure 3. The phase of transition for the initiative of adaptations.

Figure 4 illustrates that the learnt rule set from the context history (about 10 rows) at noon on 25th June is very simple, whereas the learnt rule set from the context history (about 30 rows) in the evening on the same day is more complicated. The number of rows in the context history is dependent on the weather condition of the day, season, and the location of office, because these factors can largely influence on the frequency of context changes (especially, for the temperature). Therefore, the time took for proper learning would be different according to the time and place for this kind of experiment.

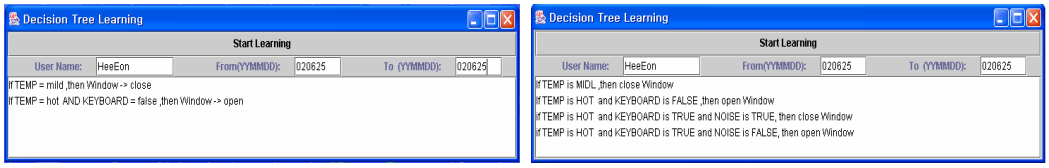

Figure 4. Learnt rule sets from different size of context history

3. The Sources of Uncertainty and Countermeasures

By analysing our approach towards proactive adaptation we can identify four different sources of uncertainty. These different sources of uncertainty and possible countermeasures are discussed in the following sub-sections.

3.1 Sensors

Context is sensed under uncertainty because physical sensors can produce erroneous or at least inaccurate data. In our experiment, we use a DrDAQ data logger from Pico Technology (figure 5) in order to obtain the state of the user's office environmental. The DrDAQ data logger has built in sensors for detecting various environmental contexts (e.g. light level, sound level and temperature) and optional external sensors. The accuracy of temperature sensed by the DrDAQ is 2°C at around 25°C (the error margin is different depending on the temperature). If the threshold between "hot" and "mild" is 25°C then this 2°C error margin could clearly be very significant (i.e., under the same (real) temperature, it could be sensed either "hot" or "mild").

Figure 5. A DrDAQ data logger.

If a context must be obtained concretely in a certain environment, more reliable value of context can be obtained by obtaining context concurrently from more than one sensor. For example, if the temperature in an emergency room of a hospital must be concretely captured, we can use several sensors, for example, one on the ceiling, one on the wall, and one on the bed. Then, using a simple majority voting algorithm we can obtain the temperature of the emergency room with reasonable reliably.

3.2 Discretisation of continuous-valued attributes

Our current approach is to use a decision tree algorithm for inducing rules from context history that can be used to provide the user with an explicit and intelligible explanation for the system's proactive behaviour. A learning algorithm based on, for example, neural networks would have been much less suitable because the weights produced by this approach are difficult to interpret by human users [14].

In general, decision tree algorithms use nominal values of context, however the DrDAQ data logger provides continuous values. Therefore, it was necessary to explicitly convert numeric context values (e.g., 26°C) to symbolic ones (e.g., hot). In this way, continuous-valued contexts are discretised by partitioning the range of context values into sub-ranges (e.g., 'hot', 'mild' and 'cold' for temperature; 'low', 'normal' and 'high' for humidity; and 'dim', 'normal' and 'bright' for light level). One implication of discretisation of continuous context values is that the value of a context can vibrate around a threshold. For example,

when the temperature is fluctuating around the threshold (24°C) between 'hot' (e.g., 24.1°C) and 'mild,' (e.g., 23.9°C) proactive adaptations can occur frequently, which can be very frustrating to the user.

A possible way to overcome these potential problems is to adopt a fuzzy representation of context [13] and utilise fuzzy decision trees [10] [11] [16] [18]. For example, given that the thresholds for partitioning the range of temperature into sub-ranges are: cold < 20°C, 20°C ≤ mild ≤ 25°C", and hot > 25°C, then figure 6 illustrates the temperature ranges using the conventional representation ([a] in figure 6) and the fuzzy representation ([b] in figure 6). The horizontal axis represents the temperature values and the vertical indicates membership values of fuzzy sets. Membership value indicates the degree (from 0 to 1) to which a given input belongs to a fuzzy set. For example, the temperature 18°C is converted into 'cold' with membership value 1 (or 100%), 'mild' with 0%, and 'hot' with 0% but the temperature 20°C is converted into 'cold' with membership value 0.5 (or 50%) and 'mild' with 50%, and 'hot' with 0% according to the fuzzy representations in figure 6.

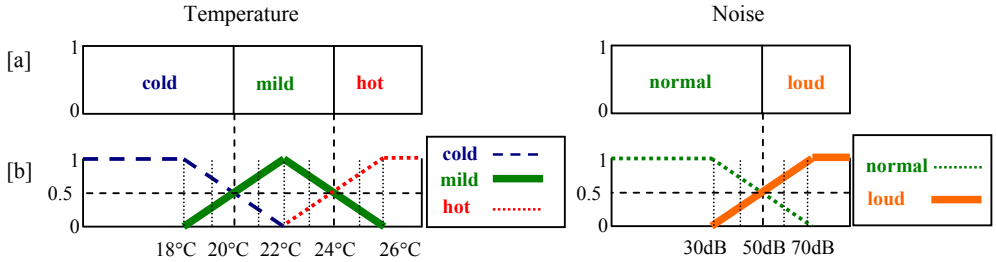

Figure 6. Conventional and fuzzy representation of temperature and noise ranges.

Next, these membership values as well as information gains are considered in order to build a fuzzy decision tree. Let's assume that a decision tree is constructed as depicted in figure 7, the temperature sensed is 25°C and the noise level is 70dB. In the conventional decision tree, this situation is just classified as the class 'Close' from the root (Temperature) and the suggestion "shall I close the window?" will be made. In the fuzzy decision tree, the left node ('hot') of Temperature has membership value of 0.75 and right node ('mild') has value of 0.25 according to the fuzzy representation in figure 6. Next, the left node ('loud') of Noise has 1.0 and the right node ('normal') has 0.0.

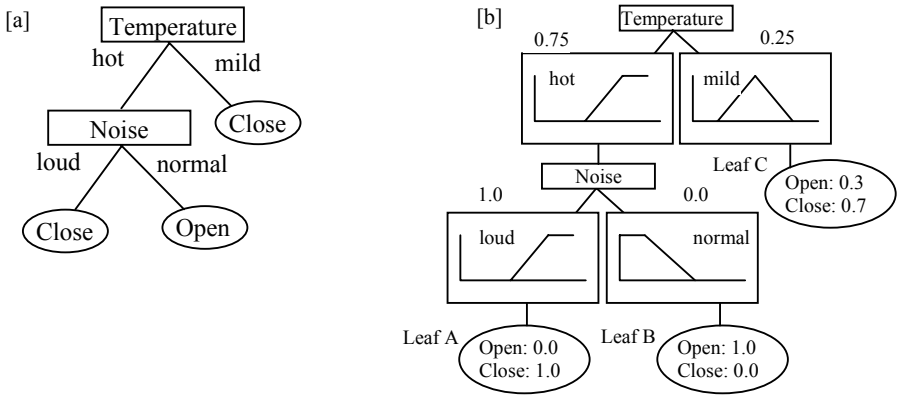

Figure 7. [a] a conventional decision tree and [b] a fuzzy decision tree
- given Temperature of 25 degrees and a Noise level of 70 decibels

Finally, we can calculate the value of each class (Open, Close) in each leaf node (Leaf A, Leaf B, and Leaf C) as follows:

- Leaf A: the membership value of "open" = 0.0(node) * 1.0(noise) * 0.75(temperature) = 0.0
- Leaf A: the membership value of "close" = 1.0(node) * 1.0(noise) * 0.75(temperature) = 0.75
- Leaf B: the membership value of "open" = 1.0(node) * 0.0(noise) * 0.75(temperature) = 0.0
- Leaf B: the membership value of "close" = 0.0(node) * 0.0(noise) * 0.75(temperature) = 0.0
- Leaf C: the membership value of "open" = 0.3(node) * 0.25(temperature) = 0.075
- Leaf C: the membership value of "close" = 0.7(node) * 0.25(temperature) = 0.175

■ The total membership value of "open" = 0.0(leaf A) + 0.0(leaf B) + 0.075(leaf C) = 0.075
■ The total membership value of "close" = 0.75(leaf A) + 0.0(leaf B) + 0.175(leaf C) = 0.925

Note that each leaf node (the nominal value of the target function: 'Open' or 'Close') has a membership value that stems from the membership values of context history data. In order to choose the value of the target function, the membership values of leaf nodes and the membership values of current contexts are multiplied and added for the same target values. Hence, the target value that has the largest one is selected as a suggestion. It is also important to note that because the result from the calculation is also a membership value of the target value (in the above case, 0.925 for 'close') this value indicates the level of certainty of the suggestion.

Staying with this scenario, if the temperature in the office was sensed at 24 degrees, the membership value of 'hot' would change to 0.5 and that of 'mild' would change to 0.5. Therefore the certainty level of 'close' would be 0.85 based on the above calculation procedure. Again if the temperature in the office was sensed at 23 degrees, the membership value of 'hot' would change to 0.25 and that of 'mild' would change to 0.75. Therefore, the certainty level of 'close' would be (leaf A) 0.25 + (leaf C) 0.525 = 0.775. The membership value of leaf C is greater than that of leaf A. This means that mild temperature is the main reason for closing window rather than the noise level. For more detailed information on the fuzzy decision tree used in this section, please refer to [18].

To sum up, firstly, fuzzy representation of context can help reduce the problem of fluctuation (e.g., the temperature's fluctuating around the threshold (25°C) between 'hot' (24.1°C) and 'mild,' (23.9°C)). In the traditional discretisation, the two temperatures (24.1 °C and 23.9°C) are converted into totally different nominal values ('hot' and 'mild') even though the differences are just 0.2°C. However, in the fuzzy handling, every nominal value of an attribute has its membership value, for example, the temperature 24.1 °C is converted into 'hot' with 52.5%, 'mild' with 47.5%, and cold with 0% (not just one nominal value). The temperature change from 24.1 °C to 23.9°C or vice versa would change the membership value of the target value as illustrated in the above paragraph. Therefore, frequent adaptations caused by the fluctuation around a specific context value can be avoided; instead the degree of certainty would be changed. Setting the certainty threshold to an appropriate value can also be used to reduce the extent to which fluctuations around a specific context value are likely to occur. For illustration purposes, consider the following (based on the fuzzy representation shown in figure 6 [b]).

1) When Temperature is 25°C and Noise is 30dB: Open: 0.825, Close: 0.175
2) When Temperature is 24°C and Noise is 30dB: Open: 0.65, Close: 0.35
3) When Temperature is 23°C and Noise is 30dB: Open: 0.475, Close: 0.525
4) When Temperature is 22°C and Noise is 30dB: Open: 0.3, Close: 0.7

In the table1, with the threshold 0.5, the suggestion changes from 'Open' to 'Close' when the temperature changes from 24°C to 23°C (just 1°C difference), whereas with the threshold 0.7, the suggestion changes when the temperature changes from 25°C to 22°C (3°C difference). Therefore, higher value of threshold can avoid inappropriate frequent suggestions.

Time		T1	T2	T3	T4
Temperature		25°C (Hot: 0.75 Mild: 0.25)	24°C (Hot: 0.5 Mild: 0.5)	23°C (Hot: 0.25 Mild: 0.75)	22°C (Hot: 0.0 Mild: 1.0)
Noise Level		30dB (Normal: 1.0)	30dB (Normal: 1.0)	30dB (Normal: 1.0)	30dB (Normal 1.0)
Threshold 0.5	Suggestion when window state is open	No Suggestion	No Suggestion	**Close**	Close
	Suggestion when window state is closed	Open	**Open**	No Suggestion	No Suggestion
Threshold 0.7	Suggestion when window state is open	No Suggestion	No Suggestion	No Suggestion	**Close**
	Suggestion when window state is closed	**Open**	No Suggestion	No Suggestion	No Suggestion

Table 1. The differences in suggestion timing between two thresholds.

However, one implication of increasing the threshold is that it can effectively cause an extended delay before an adaptation takes place. Table 2, (which is effectively a subset of the information contained in table 1) highlights how increasing the threshold from 0.5 to 0.7 causes the adaptation to be triggered at t4 instead of t3.

Time		T1	T2	T3	T4
Temperature		25°C (Hot: 0.75 Mild: 0.25)	24°C (Hot: 0.5 Mild: 0.5)	23°C (Hot: 0.25 Mild: 0.75)	22°C (Hot: 0.0 Mild: 1.0)
Noise Level		30dB (Normal: 1.0)	30dB (Normal: 1.0)	30dB (Normal: 1.0)	30dB (Normal: 1.0)
Threshold 0.5	(Assuming window state at T1 is OPEN)	No Suggestion	No Suggestion	**Close**	No Suggestion
Threshold 0.7	(Assuming window state at T1 is OPEN)	No Suggestion	No Suggestion	No Suggestion	**Close**

Table 2. Example implication for delayed adaptation when increasing thresholds.

3.3 Rule extraction from context history

The rules extracted from context history also hold uncertainty, because context history may be incomplete as a training data set and the learning method adopted may not be a perfect one. During our experimentation, we empirically learned that the size of training data can affect the accuracy or reliability of learning. For example, at the initial stage of learning in our experiment (from t0 to t1 in figure 3) the context history was not large enough for extracting appropriate rules. Our system could provide more refined suggestions as time went by (from t1 in figure 3). Therefore, we can say that more training data the better the resilience to incomplete data. However, large size of context history will increase the cost for processing learning algorithms and for maintaining storages. This trade-off relation between small size and large size of context history is outlined in figure 8. In addition, learning from a part of context history that is related to an unusual situation (e.g., the case of high noise level) may generate a decision tree that can effectively consider the specific situation. As a result, more certain proactive adaptations for the unusual case can be provided.

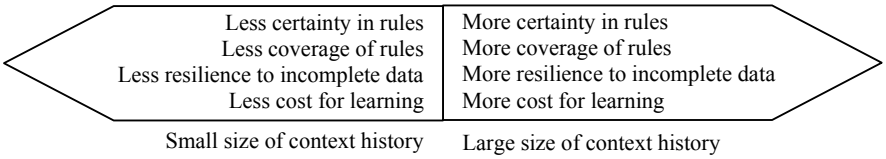

Figure 8. The characteristics of learning system based on the size of context history.

4. Interaction Issues

In addition to the uncertainty issues discussed in the previous section, there are interaction issues to be considered as follows:

- How can the user give a feedback to the system and how can the feedback be effectively considered within the process of decision making?
- How can the user effectively amend her user models (i.e., the extracted rules)?
- How can the level of uncertainty of proactive adaptations be represented to the user?

In our experiment, we let the user override a proactive suggestion by just clicking “No” button on the user interface (figure 9a) and provide an explicit explanation for the suggestion if the user clicks “Why” button (figure 9b). Given that the user clicks “No” button against the suggestion of “shall I open the window for you?” this means either the user does not accept the suggestion only in a specific time or the user want to override the rule itself that generates the suggestion. In this case, we removed the rule from the set of rules during the day but it could be generated again in the next learning stage of our experiment. Therefore, it may be more suitable to make two groups of rules: one is ready to be applied and the other is overridden by the user, and the system can check the rule is in the first group or second group before giving a suggestion. As another possible solution, by enabling the user to inspect the context history and

the uncertainties involved the user can manually remove the context with high uncertainty or which clearly represents an exceptional case.

Figure 9a. A suggestion window.

Figure 9b. An explanation window.

There is a growing concern for representing uncertainty to users. Membership values of the final classification using a fuzzy decision tree may be used as a way for representing uncertainty of suggestions to the user as described in section 3.3. In order to let the user override certain rules being created by the decision tree, the system must allow the user to observe which rules have the most or least corroborating evidence, i.e. from the context history. Because some rules will have more or less certainty than others it makes sense to make this transparent to the user in some way in order to enable the user to override uncertain rules.

5. Summary

This paper has described our exploration into the issues of uncertainty inherent in proactive and tailored behaviours under ubiquitous computing environments. Through our previous works that lead to this investigation, we found a number of sources of uncertainty and investigated countermeasures that can mitigate the level of uncertainties. In addition, we have discussed interaction issues between the system and users under uncertainty. In addition, we need a way of representing uncertainty to a user so that the user can observe which rules have the most or less certainty than others.

To summarise, through our investigation, the following key issues arise:

- Through initial analysis, it appears that applying fuzzy representation of context and fuzzy decision trees is one possible solution for overcoming the limitations of discretisation of continuous-valued contexts.
- The membership value resulted from the classification using a fuzzy decision tree can be used as a way for representing uncertainty of suggestions to the user.
- We have also determined that utilising appropriately large size of context history can provide more certain rules, more coverage of rules, and more resilience to incomplete data.
- By providing explicit explanations regarding the level of uncertainty for proactive adaptations, the user can override the adaptation and even amend the rules and/or context history.

Acknowledgements

The work described in this paper was partly supported by the Future and Emerging Technologies programme of the Commission of the European Union under research contract IST-2000-26031 (CORTEX - CO-operating Real-time senTient objects: architecture and EXperimental evaluation).

Reference

1. Abowd G.D. and E. D. Mynatt (2000) Charting Past, Present and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction, Special issue on HCI in the new Millennium, v7(1).
2. Byun H.E. and K. Cheverst (2001) Exploiting User Models and Context-Awareness to Support Personal Daily Activities, Workshop in UM2001 on User Modelling for Context-Aware Applications, Sonthofen, Germany.
3. Byun H.E. and K. Cheverst (2002) Harnessing Context to Support Proactive Behaviours, Proc. Workshop in ECAI2002 on AI in Mobile Systems (AIMS2002), Lyon, France.
4. Chen, D., Schmidt, A., Gellersen, H.W. (1999) "An Architecture for Multi-Sensor Fusion in Mobile Environments", In Proceedings International Conference on Information Fusion, Sunnyvale, CA, USA.
5. Cheverst K., N. Davies, K. Mitchell and P. Smith (2000) Providing Tailored (Context-Aware) Information to City Visitors, Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Trento.
6. Coen M. (1998) Design Principles for Intelligent Environments, AAAI'98, Madison, WI.

7. Dey A.K. and G.D. Abowd (2000) The Context Toolkit: Aiding the Development of Context-Enabled Applications. Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland.
8. Dey A.K. and G.D. Abowd (2000) CyberMinder: A Context-Aware System for Supporting Reminders, Symposium on Handheld and Ubiquitous Computing, Bristol, UK.
9. Dong M. and R. Kothari (2001) Look-Ahead Based Fuzzy Decision Tree Induction, IEEE Transactions on Fuzzy Systems, vol 9 (3), pp 461-468.
10. Guetova M., S. Holldobler and H. Storr (2002) Incremental Fuzzy Decision Trees, Proc. Of the 25th German Conference on Artificial Intelligence (KI2002).
11. Janikow C. Z. (1996) Exemplar Learning in Fuzzy Decision Trees, In Proc. Of FUZZIEEE, pp 1500-1505.
12. Lamming M. and M. Flynn (1994) "Forget-me-not" Intimate Computing in Support of Human Memory, Symposium on Next Generation Human Interface, Meguro Gajoen, Japan.
13. Mantyjarvi J. and T. Seppanen (2002) Adapting Applications in Mobile Terminals Using Fuzzy Context Information, Proc. Fourth International Symposium on Mobile HCI2002, Pisa, Italy.
14. Mitchell T. M., R. Caruana, D. Freitag, J. McDermott and D. Zabowski (1994) Experience With a Learning Personal Assistant, Communications of the ACM, 37(7).
15. Pascoe, J. (1998) "Adding Generic Contextual Capabilities to Wearable Computers", In the Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98), pp. 92-99, Pittsburgh, PA, IEEE. October 19-20.
16. Shiu S. C. K., C. H. Sun, X. Z. Wang and D. S. Yeung (2000) Maintaining Case-Based Reasoning System Using Fuzzy Decision Trees, In Proc. Of the 5th European Workshop on Case-Based Reasoning, Berlin.
17. Weiser M. (1993) Some Computer Science Issues in Ubiquitous Computing, Communications of the ACM, Vol 36(7).
18. Zeidler J. and M. Schlosser (1996) Continuous-Valued Attributes in Fuzzy Decision Trees, Proc. Of the 6th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp 395-400, Granada, Spain.

A Specialized Representation for Ubiquitous Computing and User Modeling

Dominik.Heckmann@dfki.de
European Post-Graduate College Language Technology and Cognitive Systems
Saarland University

Abstract

Ubiquitous computing offers new chances and challenges to the field of user modeling. With the markup languages UserML and its corresponding ontology UserOL we try to contribute a platform for the communication about partial user models in a ubiquitous computing environment, where all different kinds of systems work together to satisfy the user's needs. The expected result is that "permanent evaluation of user behaviour with different systems and devices will lead to better user models and thus allow better functions of adaptation like adaptive websites, recommended products, adaptive route planning or better speech interaction. We also present an implementation of a general user model browser and editor which is based on UserML and which is embedded in an ubiquitous computing simulation environment with the name UBI's WORLD, see [6].

The keywords are ubiquitous computing, distributed user modeling and markup languages.

1 Introduction

In 1995, Jon Orwant already claimed in his Doppelgänger [2] project: *We need a protocol for encoding information about users, any given user modeling system should be able to benefit from others and that user models should follow you around.* If we look at ubiquitous computing through the eyes of user modeling and decide to enable the interaction to be uniformly user-adaptive, a need for extended user model communication becomes obvious.
This work is highly under progress and not in any stable state yet, but two of the main research questions on which we are working are *Qestion1: How can a user modeling language be designed, in order to enable the communication with different user modeling applications ubiquitously?* and *Qestion2: How can the users inspect and control their user models?*
We are developing the XML-based exchange language `UserML` (see e.g. [4]) which is based on an ontology that defines the semantics of the XML vocabulary (UserOL). This user modeling ontology language is based on a taxonomy from Jameson and Kobsa (see e.g. [3]).
A gerenal tool is currently implemented that allows the user to inspect and change his/her user model. UserML sentences will be transformed with XSLT into the W3C candidate recommendation XForms that can be viewed on any devices.
Let us assume people carry a personalized user-modeling agent within a PDA and transmit their long-term properties anonymously to human-computer interaction systems. This could enable user-adaption from the beginning of using a new interaction system. After the interaction session, the system could transmit the partial user model of this session back to the personal agent within the PDA and also use the enriched data anonymously for collaborative filtering for example.

2 The User Modeling Markup Language UserML

Using XML as knowledge representation language has the advantage that it can be used directly in the Internet environment. For UserML we have chosen to take a modularized approach in which several modules will

be connected via identifiers and references to identifiers. With this method, the tree structure of XML can be extended to represent graph structures. The content of a `UserML` document will be divided into `MetaData`, `UserModel`, `InferenceExplanations` as well as `ContextModel` and `EnvironmentModel`. The main focus in this workshop contribution lays on the UserModel but however, in order to explain inferred user model entries, the instrumented environment and the context (see i.e. [3]) of the interaction process need to be represented or referred to as well.

2.1 Example: A User-Adaptive Airport Navigation System

In this subsection we present an example of a user-adaptive airport navigation system, which is currently under development in the integrating scenario of the *German Collaborative Research Center on Resource-Adaptive Cognitive Processes, SFB378, in the projects READY and REAL.*
How can we represent for example the following derived user property together with the description of the situation in UserML? *A system at an airport detects that a person is currently under high time pressure because she has a flight ticket for a flight, which boarding time will probably close in 10 minutes and the user still has to navigate to the gate.*

2.2 The UserModel Syntax

The approach that we suggest separates between two different levels. On the first level, we offer a simple XML structure for all entries of the partial user model. These `UserData` elements consist of the elements: `category`, `range` and `value`. On the second level, we find the ontology that defines the categories. The advantage of this approach is that different ontologies can be used with the same UserML tools. Thus different user modeling applications could use the same framework and keep their individual user model elements. The reference to the ontology like "http://www.u2m.org/UserOL/" can either be placed in the meta data module, where it will serve as a default value for all UserData entries, or in the UserData entry itself.

Example of a partial user model which uses categories from the ontology "UserOL"

```
<UserModel>

  <UserData id="231">
    <category>userproperty.timepressure</category>
    <range>low-medium-high</range>
    <value>high</value>
  </UserData>

  <UserData id="224">
    <category>userproperty.walkingspeed</category>
    <range>slow-normal-fast</range>
    <value>fast</value>
  </UserData>

  <UserData id="122">
    <category>usercontext.location</category>
    <range>airport.location</range>
    <value>X35Y12</value>
  </UserData>

</UserModel>
```

The UserModel consists of an unbounded list of UserData entries. Each one defines the category, the range and the value. The alternative approach would have been to encode the user modeling knowledge into the XML elements like `<timepressure>`, `<psychologycal-states>`, `<typing-behaviour>`. Confidence values, a notion of time and the references to the inference explanations will be added with the next step.

A Preliminary DTD for the Element "UserModel"

```
<!ELEMENT  UserModel (UserData)*>
<!ELEMENT  UserData (category,
           range, value, ontology?)>
<!ATTLIST  UserData id ID>
<!ELEMENT  category (#PCDATA)>
<!ELEMENT  range (#PCDATA)>
<!ELEMENT  value (#PCDATA)>
<!ELEMENT  ontology (#PCDATA)>
```

Since the user model ontology and the markup language are still under construction, their progress will be presented at the webpage `http://www.u2m.org/UserOL/` and `http://www.u2m.org/UserML/`.

3 A User Model Editor

If a system collects user data, the person should have the possibility to inspect and edit this

model in a human readable format. A private (not necessarily mobile) device seems to be a good choice to serve as an editing tool for user models of different user-adaptive systems. Especially in ubiquitous computing not every user-adaptive system will have a user interface. With the help of UserML the data and the user model editor could be send to the nearest user interface.
There are currently two implementations under construction. One transforms UserML into the W3C Candidate Recommendation "XForms" (see [5]) with XSLT. XForms documents can be interpreted by web browsers or on mobile devices with a java vm. And the second one uses a MySQL database and generates a web interface, which is embedded in UBI's WORLD, see [6], a tool to simulate ubiquitous computing environments.

Conclusion

We think that ubiquitous computing will have a great influence on user modeling. The development of a markup language for user modeling within the new paradigm of ubiquitous computing is important. The main idea of UserML is to enable communication about partial user models via the Internet. In this paper, we put the focus on the aspect of representing partial user models. This work is under progress.

References

[1] Marc Weiser: "The Computer for the 21st Century". Scientific American, 265(3) (1991)

[2] Jon Orwant: "Heterogeneous Learning in the Doppelgänger User Modeling System", User Modeling and User-Adapted Interaction **4** (1995)

[3] Anthony Jameson: "Modeling Both the Context and the User", Personal Technologies, **5** (2001)

[4] Dominik Heckmann: "Ubiquitous User Modeling for Situated Interaction", UM2001

[5] W3C: "XForms Candidate Recommendation", http://www.w3.org/TR/xforms/

[6] Dominik Heckmann: "Ubi's World for IE6" http://www.u2m.org/ubisworld.htm

Merino: Towards an intelligent environment architecture for multi-granularity context description

Bob Kummerfeld [a], Aaron Quigley [b], Chris Johnson [c], Rene Hexel [d]

[a] *Smart Internet Technology Research Group, School of Information Technologies, University of Sydney, Australia*

[b] *Smart Internet Technology Research Group, School of Information Technologies, University of Sydney, Australia*

[c] *Department of Computer Science, Australian National University, Canberra, Australia*

[d] *School of Computing and Information Technology, Griffith University, Nathan Campus, Australia*

Abstract

The Intelligent Environment consists of ubiquitous connectivity in sensor rich environments coupled with adaptive technologies that allow access from multiple devices with varying capabilities. Context information is drawn from what the Intelligent Environment can sense about its current physical and computational environments. This context information relates to aspects of the environment which include, who is in the environment, what they are doing, what they have done, and what their actions are. Multi-granularity context description provides a mechanism to describe, at increasingly more abstract levels, such context information gleaned from the Intelligent Environment. The proposed architecture for the Intelligent Environment is central to determining methods for capturing such information at the lowest level, interpreting by historical reference, and aggregating into increasingly more abstract forms.

Key words: Context; Personalisation ; Smart-Sensors ; Personas ; Smart Environment Agents.

1 Introduction

Ubiquitous computing requires computers to have more knowledge of the real world, the environment surrounding users, than computing did previously. The ubiquitous computing environment therefore has to handle large numbers of sensors, and to hide the physical sensors and communications under layers of abstraction. The sensor devices range in complexity, from devices as simple as a binary on-off, up to sensors that can decide "a meeting is now happening in this room". Their communications abilities range from the simple on-off binary signal to supporting Web services.

The raw data produced by the sensors has an enormous range of speed and volume – it may be as simple as a one bit value on a frequency of hours or days, up to a continuous stream of video data. To enable higher-level inferencing to be performed with sensor information, that information must be consolidated, in many cases, and must be distributed to the computing devices and programs that wish to deal with it.

Adaptive sensors and integrating sensors are therefore an essential part of bridging between the raw devices and an intelligent ubiquitous computing environment that includes user modelling[Kautz et al., 2003].

The issues include both the representation of data coming from raw sensors in a form suited to programs that are capable of reasoning with it, including sensors that have a low level of intelligence but deal directly with other sensors' information; and the efficiency, security and privacy related to transmitting the information from sensors and user models over any kind of network[Schreck, 1997,Busboom, 2002].

The scale of the ubiquitous computing system is significant to the design of both the distribution of sensor information and the way in which sensors and information are represented. To cater for even one enterprise of the scale such as a manufacturing plant, a shopping mall, or a university, it must be scalable to the extent

Email addresses: bob@it.usyd.edu.au (Bob Kummerfeld), aquigley@it.usyd.edu.au (Aaron Quigley), Chris.Johnson@cs.anu.edu.au (Chris Johnson), rh@cit.gu.edu.au (Rene Hexel).

of allowing tens or hundreds of raw sensors, in hundreds of rooms in a building, with hundreds of buildings to a campus [Schmidt and Beigl, 1998].

1.1 Layers of context

Sensors are assumed to be connected to the net and capable of communication. The sensors that connect directly to the environment (such as movement detectors, proximity detectors, light level meters, thermometers) typically produce raw data, at high frequency or at irregular intervals. The data comes as primitive bit streams or simple values. As the first stage of making sense of this data, each sensor's information must be capable of being

- integrated against its own history
- adapted from one representation to another
- combined with other sensors into richer forms of sensor information (where conflicts between the different sensors can be resolved).

For example, one of the standard motivating cases is that of detecting that a meeting is in progress in a particular room, allowing context-aware applications to modify their behavior accordingly. Such a sensor is clearly not a simple one: we will refer to it as a *smart* sensor. To approximate the human-based metrics that are used to make a decision like this, such as "number of people present" and "noise level", we might consider a location-based face-detecting and -counting sensor (a sensor that is based on a camera, but not transmitting images onto the network), combined with a noise-level sensor – in the same physical vicinity. The combination of values of these sensors coming above some preset threshold, within certain time periods, may trigger a smart sensor to indicate the information "meeting in progress". A computer application might use this sensor, but further confirm this indication for its own purposes, by inference using the daily calendar or user model of one or more of the participants, or the room-booking system of the owning organisation. Such a sensor might be used by the participants' own laptop or PDA computer applications to modify its way of handling of incoming messages, or by the room-booking system itself to confirm that the booking has in fact been taken up, or to control the lighting and heating levels and the access to other equipment of that room.

It is clearly a powerfully simplifying design decision to view sensors of both simple and complex kinds in a common framework, from the point of view of the computational elements of the ubiquitous computing system. Organising the whole system into layers of complexity is also necessary, given the scale of the number and variety of sensors involved.

A further requirement of a system on this scale is that the network must be capable of efficiently distributing the sensors' information and user model information beyond the immediate vicinity, but at the same time, the data and information flowing on the communication network must be made secure for both personal privacy and safety reasons.

1.2 Security and privacy

Even raw sensor information must be made secure as soon as it goes over any kind of network beyond the very local. Where sensors are reacting to the proximity of people and reporting their identity and their precise location, there are issues of privacy even in the relatively secure working campus (who needs to know that I am consulting a medical doctor in my lunch break?) and of personal security, where there is a possibility of stalking. Even the very lowest layers of context handling must provide security against inferencing that might reveal the location of a person at risk, or violate privacy policies by revealing the details of office workers' movements in fine detail within a suite of offices and its service areas.

Encryption is only part of the security solution: locations and personal identities may need to be obfuscated by depersonalisation and location fuzzing, incorporated at the level of simple sensors.

Pragmatic issues of implementation therefore lead us to have at least two layers. The complexity of handing data from raw sensors and making simple inferences from this data lead us to a further layering of the handling of sensor information.

2 Background

Early works in ubiquitous computing already touched the complexities of context gathering, processing, and representation as well as privacy and security issues [Weiser, 1991]. Despite this insight, most approaches were based on case studies and projects. In light of the complexity that arises from having multiple, distributed layers between the sensor and system levels at the bottom and the application layer at the top, most projects and case studies were aimed at a very narrow, vertical path through these layers. As a consequence, the techniques chosen in most approaches matched the particular problem addressed by the study and not so much the general case.

2.1 Major Case Studies and Projects

The Aware Home project [Kidd et al., 1999] is a conglomeration of projects aimed at building a context aware, intelligent home. The goal of the project is to create a home environment that is aware of its occupants' whereabouts and activities.

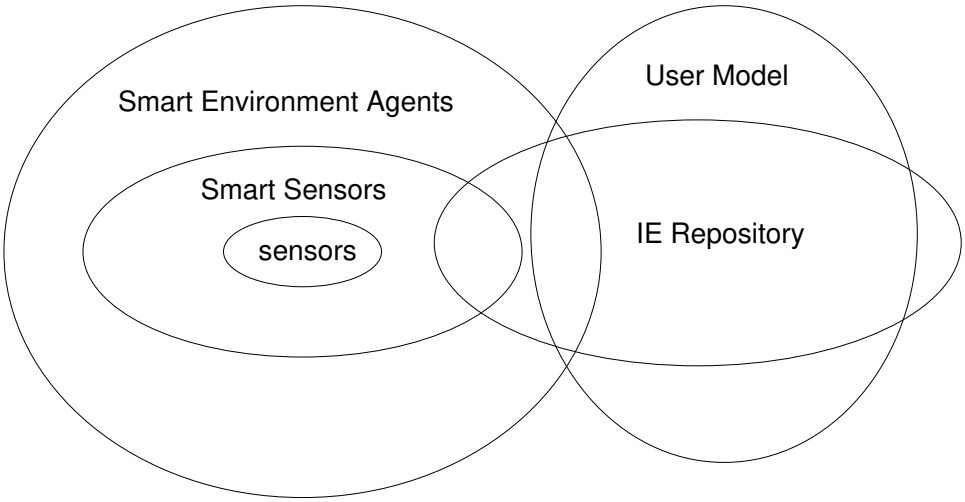

Fig. 1. Section of intelligent environment service layer description

The Massachusetts Institute of Technology houses two different context aware computing projects. Project Oxygen [Brown, 2001] is an attempt to interact with computing environments on human terms. The core emphasis of this project is to use speech and vision for user interaction. The Intelligent Room [Phillips, 1999] is an AI centred multi agent approach to embed computers into ordinary office or home environments.

Typically, these projects provide toolkits [Salber et al., 1999] or APIs that allow some form of context information to be utilised by applications. Other projects such as Cooltown [Kindberg et al., 2000] attempt to implement a loose framework that is based on the notion of a web presence for people, places, and things. It uses the existing internet and world wide web infrastructure as a glue for communication between these entities.

2.2 Context Infrastructures and Environments

An approach that uses an existing network infrastructure is the Ambient Interaction Framework [Fitzpatrick, 1998], which uses an I/O oriented framework to connect different sensors and actuators. The framework relies on a content based message routing system called Elvin [Fitzpatrick et al., 1999].

Based on their experience with the Aware Home project, Abowd et *al* presented an architecture for context aware applications that uses a widget abstraction model to represent sensors and actuators [Salber et al., 1999]. A similar architecture was presented by Grimm et *al* that uses a task/tuple abstraction model to handle context information [Grimm et al., 2000].

2.3 Context Distribution and Scalability

While the architectures presented above are suited to the well defined and constrained environments they were designed for, they lack flexibility in open, distributed environments. A first attempt to address part of this issue was presented by Kantor and Redmiles [Kantor, 2001]. Their approach is to introduce a group based distributed subscription service that allows applications to locate and subscribe to context information provided by other applications at various abstraction levels.

Despite multiple attempts to solve parts of this problem, at the moment, the biggest challenge remains scalability in large scale, dynamically changing, distributed environments. The system architecture introduced in this paper attempts to address this problem at the very foundation of sensor and actuator interaction with intelligent, user centred applications.

3 Merino Architecture: Layers of context

The proposed architecture for context layers for the sensed Intelligent Environment is shown in Figure 1. This architecture consists of five distinct elements, sensors, smart sensors, smart environment agents, the Intelligent Environment repository, and finally the user model.

At the lowest layer, the sensors are mechanisms in both hardware and software to interrogate both the physical and computational environment. These sensors include, hardware to measure the physical environment such as temperature, motion, light, vibration, and pressure, along with software sensors which detect user login, network activity, and application activity. Sensors are assumed to be net connected devices, where connectivity ranges from conventional fixed line to wireless sensor networks. Existing work in ubiquitous computing has often focussed on providing such voluminous data as *context* which an application engineer must then incorporate and interpret in an ad-hoc application specific

manner [Hawick and James, 2003,Salber et al., 1999]. In contrast, here the lowest layer of abstraction an application engineer can use is the *smart sensor layer*.

3.1 Smart sensor layer

Smart sensors, provide the first layer of context abstraction in our intelligent environment architecture, where the repository also acts as the secure and scalable distribution mechanism for context information. Rather than provide the sensor data at the lowest level of detail (raw data) to an application programmer, here the smart sensor layer is provided. The smart sensor layer includes methods for aggregating, filtering, and eliding the raw sensor data into forms which are made available through the repository interface. For example, a location context agent would harvest data, possibly contradictory, directly from a range of physical and software sensors. This data is used to determine a person's physical whereabouts, as being present in a identified location. This "location" information is then deposited into the IE repository by the context agent.

3.2 Smart environment agents

Smart environment agents provide the second layer of context abstraction. These agents form into two classes, *rich-context* providers and *performance enhancers*.

Rich-context providers, access a range of information from the IE repository to form higher level context information or "rich-context". For example, a meeting smart environment agent may access location context information, calendar context information, and noise context information to form a high level piece of rich-context information, that a meeting is taking place. In general such agents build such high level abstraction by the use of first level smart sensor information, user model information, and historical smart sensor and user model information.

Performance enhancers, are the second class of smart environment agents which act with environmental knowledge over the smart sensor layer improving their individual and collective performance. Such agents interact with both the smart sensor layer and the computing environment to determine methods of improving the performance of the global smart sensor layer. One approach is for these agents to include learning and reasoning algorithms to discover patterns which enable the agents to improve the performance (eg. maximize battery life) and scalability (eg. minimize communications) of the underlying smart sensor layer.

3.3 Rich Context

"Rich Context" information is provided on varying levels of granularity. At the highest level an application may want to only listen for "meeting" events but once this event is raised it needs to know the components of the information used to form this decision, and so on. In this way, the Intelligent Environment allows application designers to build risk-adverse context aware applications depending on the granularity of the information to be provided to the end user. Instead of your "smart phone" automatically redirecting an incoming call because it "thinks" you are in a meeting, the context information for both the sender and receiver are provided to both parties to facilitate a smart decision being made.

Smart environment agents also provide a means for the intelligent environment to update information about people using the environment as their associated context or rich-context changes. For example, as a person moves through the physical environment their associated user model is updated with the changing location information [Riché and Brebner, 2003].

3.4 Architecture: User Modelling

One class of agent assessing the Intelligent Environment (external to Figure 1), is the Smart Personal Assistant which takes care of a number of chores for the user. In practice, such Smart Personal Assistants access and manage much of the user model, along with accessing context and rich-context from the IE repository to customise and personalise interactions on behalf of the user.

4 Realisation

In our architecture, context information is stored as fields of context objects that represent particular sensors or higher level context entities. For example, an environmental sensor may have an object with fields for temperature and light level. A higher level context entity such as a room may have a list of people that are currently in the room.

Context objects are modified by sensor agents or by smart environment agents. A sensor agent may have the task of monitoring a low level environment sensor and updating a field in the relevant context object. Smart Environment Agents in turn may monitor context objects and add or modify fields of other context objects. A Smart Environment Agent may monitor sensor objects for a room and infer that several people are present and meeting is in progress. This information is stored in a context object for the room.

The context repository is implemented as a type of distributed object database. The objects are stored at a set of servers distributed across the network in administrative domains (Fig 2). Servers are arranged in a tree and can locate parent and child servers.

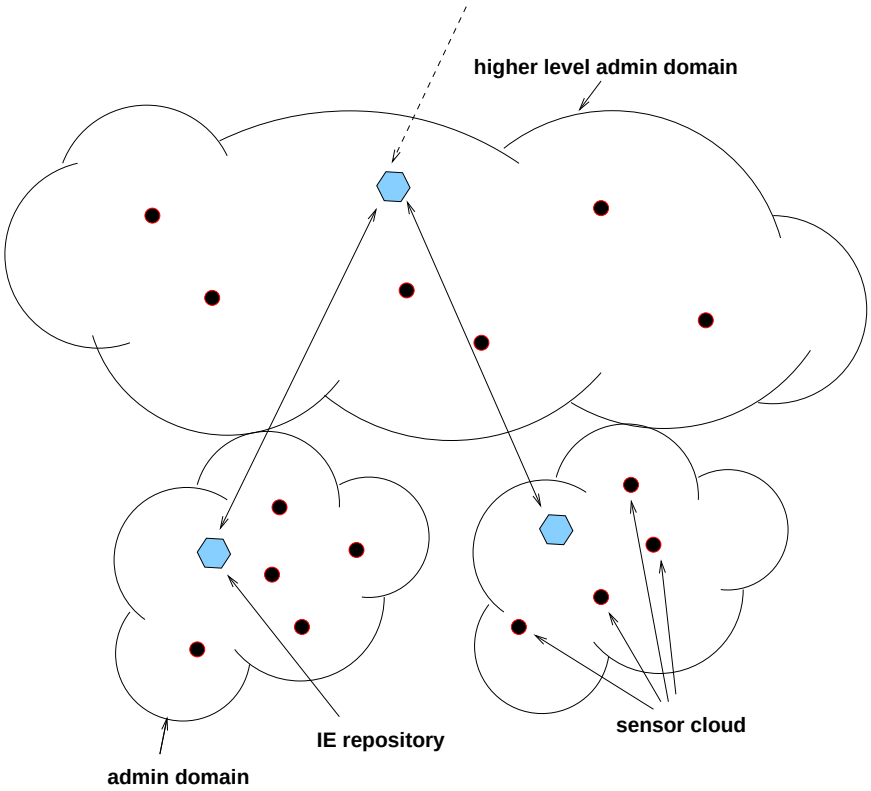

Fig. 2. Intelligent environment sensor cloud distribution and aggregation

Objects are referenced using a unique ID or URN. In order to find a given object a search is carried out starting at the local server and expanding through the tree in a similar way to Gloss [Dearle et al., 2003] and Globe [Ballintijn et al., 1999] systems. The key difference is that our search has a configurable limit by administrative domain. Beyond this limit the search switches to a distributed hash table approach. The reason for this approach is to restrict most searches to the local area where they are most likely to be satisfied and where the natural ownership of context data lies but still allowing a scalable solution that doesn't default to global flooding. Once the server for an object has been located, operations are then carried out directly with the server. Key features of the context repository are that each object has a unique ID and resides at a single point (server) in the system, and is located using a distributed search.

Communication at the local server level is carried out using a multicast message-based protocol. For our initial implementation we use the Elvin system [Fitzpatrick et al., 1999]. Communication between servers is also message-based but unicast.

Agent programs may register with a context repository server and ask to be notified when a specified event occurs such as a particular object being updated. This allows, for example, an agent to monitor a room for changes. Notification occurs using the same message protocol used in other parts of the system

For a complete example of the system in operation consider a person carrying a Bluetooth-equipped phone entering a room with a Bluetooth sensor. The sensor detects the presence of the phone and notifies its controlling agent which converts the (unique) Bluetooth ID of the phone into a URN. The agent locates the local context repository server and requests the home server for that URN. The server may be the local server for a phone that normally resides in the building but if not a search is carried out, eventually returning the server address or a not-found indication. The home server of the Bluetooth device object is then contacted and information added to the relevant object to indicate that the Bluetooth device has been detected by the sensor. Agents that have asked for notification when this object changes are then notified. Among these may be higher level agents that in turn generate further context information and update other objects which trigger notifications.

User models are stored in objects in the same distributed context database. These are complex objects that contain user profile information in the style of Personis [Kay et al., 2002]. This system includes a "view" mechanism that allows a subset of the user model to be retrieved. This view definition can be specified by the query agent, can be stored in the user model, or

is determined according to the query agent's identity. This feature allows the user model to present a set of "personas" to query agents.

Another feature of the Personis system is that values for components of the user model are calculated from evidence for particular values [Kay, 2000]. Evidence is accreted and when a value of the component is required it is calculated using an application specific resolver. This accretion approach has many advantages when used with context data: interpretation is delayed until a value is required and historical data (evidence) can then be interpreted (resolved) appropriately for an application.

An important feature of context-aware systems of the future is that the user should be able to find out what the system is doing on their behalf. Context data and associated inferencing systems must be inspectable or *scrutable*. This is a key feature of our user modelling system [Kay, 1995].

5 Summary and Current Status

In many current context systems there is a strict distinction between low level, sensor context information and high level context information created as a result of an inference activity. In our system the context repository unifies low and high level context information along with the user model.

Our architecture also unifies the user model with context information through the use of the *accretion* approach to managing possibly unreliable context data and the *views* mechanism for presenting different personae to context aware applications.

A prototype repository has been built for a single domain. Bluetooth access points are being used as simple sensors of Bluetooth devices with a sensor agent gathering location information and publishing it to the repository. Some simple applications have been developed to demonstrate tracking of people carrying Bluetooth-equipped phones in our building. Work is continuing on integration of the Personis user modelling system, inter-repository communication, and the representation of context information in standard formats.

References

[Ballintijn et al., 1999] Ballintijn, G., van Steen, M., and Tanenbaum, A. S. (1999). Exploiting location awareness for scalable location-independent object IDs. In *Fifth Annual ASCI Conference*, pages 321–328, Heijen, The Netherlands.

[Brown, 2001] Brown, E. S. (2001). Project Oxygen's new wind. *Technology Review*.

[Busboom, 2002] Busboom, A. (2002). Delivery context and privacy. In *W3C Workshop on Delivery Context*, Sophia-Antipolis, France. W3C.

[Dearle et al., 2003] Dearle, A., Kirby, G., Morrison, R., McCarthy, A., Mullen, K., Yang, Y., Connor, R., Welen, P., and Wilson, A. (2003). Architectural support for global smart spaces. In *Proceedings of the MDM'2003*, pages 153–164, Melbourne, Australia. Springer-Verlag.

[Fitzpatrick, 1998] Fitzpatrick, G. (1998). *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD thesis, Department of Computer Science and Electrical Engineering, The University of Queensland.

[Fitzpatrick et al., 1999] Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T., and Segall, B. (1999). Instrumenting the workaday world with Elvin. In *Proceedings of the ECSCW'99*, pages 431–451, Copenhagen, Denmark. Kluwer Academic Publishers.

[Grimm et al., 2000] Grimm, R., Anderson, T., Bershad, B., and Wetherall, D. (2000). A system architecture for pervasive computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, Kolding, Denmark. ACM, SIGOPS 2000.

[Hawick and James, 2003] Hawick, K. A. and James, H. A. (2003). Middleware for context sensitive mobile applications. In Chris Johnson, P. M. and Steketee, C., editors, *Proceedings of the Australasian Information Security Workshop and the Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing*, volume 21 of *Conferences in Research and Practice in Information Technology*, pages 133–142, Adelaide, South Australia. Computer Science Association, Australian Computer Society.

[Kantor, 2001] Kantor, M. (2001). *Creating and Infrastructure for Ubiquitous Awareness*. PhD thesis, University of California, Irvine, California, USA.

[Kautz et al., 2003] Kautz, H., Etzioni, O., Fox, D., Weld, D., and Shastri, L. (2003). Foundations of assisted cognition systems. Technical Report CSE-03-AC-01, Dept of Computer Science & Engineering, University of Washington.

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

[Kay, 2000] Kay, J. (2000). Accretion representation for scrutable student modelling. In *Proceedings of Intelligent Teaching Systems 2000*, pages 514–523. Springer-Verlag.

[Kay et al., 2002] Kay, J., Kummerfeld, B., and Lauder, P. (2002). Personis: a server for user models. In *Proceedings of Adaptive Hypertext 2002*, pages 203–212. Springer-Verlag.

[Kidd et al., 1999] Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacInyre, B., Mynatt, E., Starner, T. E., and Newstetter, W. (1999). The Aware Home: a living laboratory for ubiqutous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings*, Pittsburgh, USA.

[Kindberg et al., 2000] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., and Serra, B. (2000). People, places, things: Web presence for the real world. Technical Report HPL-2000-16, HPLabs, http://www.cooltown.hp.com/.

[Phillips, 1999] Phillips, B. (1999). Metaglue: A programming language for multi-agent systems. Master's thesis, MIT, Cambridge, MA.

[Riché and Brebner, 2003] Riché, S. and Brebner, G. (2003). Storing and accessing user context. In *Proceedings of the MDM'2003*, pages 1–12, Melbourne, Australia. Springer-Verlag.

[Salber et al., 1999] Salber, D., Dey, A. K., Orr, R. J., and Abowd, G. D. (1999). The Context Toolkit: aiding the

development of context-enabled applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems*, pages 434–441, Pittsburgh, PA.

[Schmidt and Beigl, 1998] Schmidt, A. and Beigl, M. (1998). There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces.

[Schreck, 1997] Schreck, J. (1997). Security and privacy issues in user modeling. In *UM97: The Sixth International Conference on User Modeling*.

[Weiser, 1991] Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, 265(30):94–104.

A Personal Agent Supporting Ubiquitous Interaction

B. De Carolis, S. Pizzutilo, I. Palmisano and A. Cavalluzzi
Intelligent Interfaces, Department of Informatics,
University of Bari, Italy
{decarolis, pizzutilo}@di.uniba.it

1. Introduction

There are many different ways in which context information can be used to make computer systems and applications more user friendly, flexible and adaptive especially in ubiquitous and intimate computing where the interaction situation and usage needs change rapidly. Context-awareness then refers to the ability of a system of extracting, interpreting and using context information intelligently in order to adapt its functionality to the current context of use [3,6]. Considering the effects on the interaction between the user and ubiquitous context-aware systems, there are at least two aspects that are worth mentioning: context-aware **information presentation** and **service fruition**. As far as the first aspect is concerned, results of information services could be adapted not only to more static user features such as user's background knowledge, preferences, sex, and so on, but also to more dynamic context related features such as user's activity, location, affective state and so on [1].
The second aspect regards execution of user tasks triggered by context information. For instance user's tasks present in a to-do-list or agenda could be proactively executed when the user enters in an environment or is in a situation in which those task can be executed [2,6,7]. Moreover, their execution can be contextualized according to available resources, location and so on.
Then, a system supporting these two features, taking advantage from both user and context modeling, would represent a good solution for achieving effective ubiquitous interaction.
A way to approach this problem is to take inspiration from the personal interface agents research area [18,20]. In this paradigm, the user may delegate tasks to a personal agent that may operate directly on the application or may act in the background while the user is doing something else. An agent is, in this case, a representative of the user and acts on his/her behalf more or less autonomously.
Moreover, it has to be able to communicate to the user in an appropriate way without being too much intrusive according to the context situation, user preferences, habits and needs.
The *Dream Agent* definition could be the one reported in the following sentence taken by a discussion with Cristiano Castelfranchi [10]about these issues: "It is not enough that they do what he (the master) orders, but they must understand what he desires, and frequently, to satisfy him, they must even anticipate his thoughts. For them it is not enough to obey him, they have to please him..... They have to put attention to his words, to his voice and his eyes ... completely outstretched to catch his wants and to guess his thoughts. Can you call this "life"!" (E'tienne DeLa Boétie "Discours de la servitude volontaire", 1595).
Then, hard life for our ubiquitous personal agents: helping without bothering too much, doing exactly what the user expects them to do successfully in **that context**.
Our work represents a first step in this direction. D-Me (Digital-Me) is multiagent system including two interacting entities: a D-Me Agent, representing the user, and the Environment, a physical or logical place in which various services are available. In this paper we will illustrate how D-Me supports personalized ubiquitous interaction. In order to show how the system works, we developed, as a first prototype, **My-DIB** that aims at modeling the Department of Informatics Bari (DIB) as a smart environment. Its users, that can be represented by D-Me agents, are students with different level of experience, professors, staff.
Let us assume, as a scenario example, that the user is a second-year female student, travelling each day to go to the Campus. Let's suppose that her To-Do-List includes the two following tasks:
"*Give back to the library the book on HTML*" (with low priority) and "*Find documentation for the Web Programming exam*" (with high priority).
In this scenario, My-DIB and the student D-Me would behave in the following way:

> *When the user comes close to the library, the student D-Me generates a reminder for giving back the book; it also proactively asks the library Service Agent whether Web Programming books are available and displays information received according to students rating. If it detects other student*

D-Mes with the same task it will contact them asking for more information. According to the user preferences will, eventually, put her in contact with the others.

Let's see how the system deals with this situation.

2. D-Me: a Context-Aware Agent

As stated in Dey [4], a user may benefit of a context-aware application for the proactive execution of tasks that is scheduled, for instance, in a To-Do-List. In this case, the user lists the task to be performed in different contexts and environments and gives to her agent the autonomy to perform them entirely or in part. When the user is in a particular situation (environment, time, location, emotional state, etc.) that triggers one of these tasks, her agent requests personalized execution of the services necessary to accomplish it, by transferring to the environment the information it needs. When the task has been performed, the agent communicates results to the user. These may be of various nature, according to the performed service, and can be adapted to "user in context" features. For instance, a "remind message" is generated when some user task can be performed by the user in that context; "user/context adapted information presentation" is generated after an information retrieval task; "assistance on task execution" is generated when help is needed in performing actions in the real or digital world, and so on.

According to this vision, a D-Me Agent is a context-aware agent that uses several knowledge sources to assist its "owner-user" in interacting with smart environments according to an user "to-do-list". This list contains whatever has been explicitly requested by the user or inferred by the agent according to its level of autonomy. In particular, D-Me is a BDI agent [21], since its reasoning mechanism aims at satisfying, triggering the opportune intentions, its desires (goals) by taking into account its belief (mental state).

At this stage, we modelled and implemented the reasoning behaviours aiming at achieving the following macro-desires:

- *execute totally or in part tasks specified in the user to-do-list*: this desire is quite complex and it is achieved by accessing the specification of task to be executed with the associated constraints in the user to-do-list and executing the correspondent task model.
- *create new tasks if required*: sometimes the context triggers the execution of tasks that were not explicitly stated in the to-do-list. In this case this desire become active.
- *get user-related information relevant for adapting task execution*: in order to adapt task execution and to communicate results to the user appropriately, the agent needs to know information about the user. These information can be stored in a user profile or can be inferred.
- *get context-related information relevant for contextualizing task execution*: as for user related data, assessing the current context situation is important especially for triggering and adapting task execution.
- *communicate personalized results*: results of tasks can be of various nature (information presentation, reminders, notifications, and so on). The way in which the agent communicates to the user is adapted to user interests, knowledge, preference and so on, but also to context features.

In order to achieve these goals, D-Me Agent exploits the following knowledge sources that correspond to ontologies used for inter-agent communication: i)the ***to-do-list***, containing the description of the task and its constraints in terms of activation condition and priority; ii) the ***formal description of the task***, in the form of a extended XDM-models [8], that the agent can use in order to execute it; iii) the ***Mobile User Profile*** (MUP), containing information about the user, and the iv) **context situation**.

On the other side, as shown in Fig1, the **environment** is 'active' [19]: it is populated by several D-Me Agents and by **Service Agents** which execute various tasks. While the number of Service Agents depends on how many tasks the environment supports, there is only one "Keeper Agent" that acts as a Facilitator [12] and knows which (D-Me and Service) agents populate the environment. It therefore provides to D-Me environment-related information and the list of agents which could accomplish the required service/s.

Users may interact with services in a remote way or by being physically in the environment and may then move from one environment to another. Managing inter-environment communication is the task of the **EnvKeeper** that allows to handle a federation of environments.

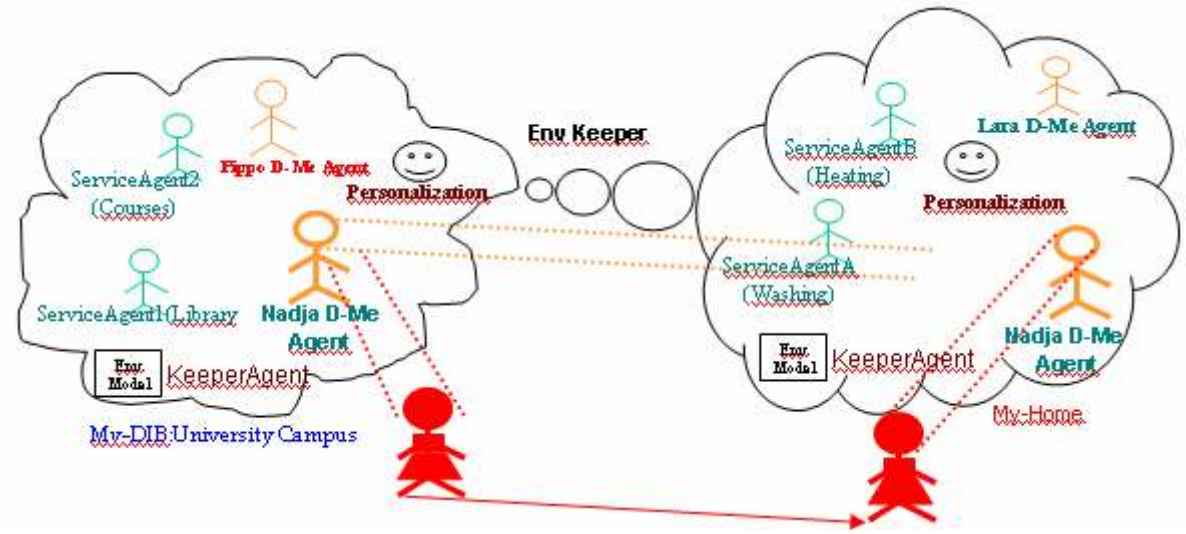

Fig.1. Overview of the D-Me model

The D-Me architecture has been developed using the JADE toolkit [14] which is FIPA [11] compliant. Communication among agents has been approached developing specific protocols and using FIPA Agent Communication Language (ACL) compliant messages. The only constraint is that every entity that supports this communication must share with the other parties a portion of an ontology [13], whose concepts are the subject of the conversation.

2.1 Supporting Ubiquitous Personalization

The need to let the user free to interact with services everywhere and continuously in time causes obvious changes in the way the personalization component has to be designed and developed [2,15,16,17]. In this optics, the following problems and challenges arise:
a) *Location*, when services are ubiquitous, users can access them from everywhere and continuously in time, moving physically in different environments, this requires the management of different strategies for locating and accessing information about the user (user profiles).
b) *Security*, information about the user is not always used by the same environment, but it "moves" with its "owner"; this requires the need of establishing privacy and security policies.
c) *Consistency*, as mentioned before user may interact with more than one environment at a time; then, it is necessary to develop a strategy for keeping individual user information consistent.
Possible approaches to these problems are represented by a centralized, distributed or mobile approach [17].

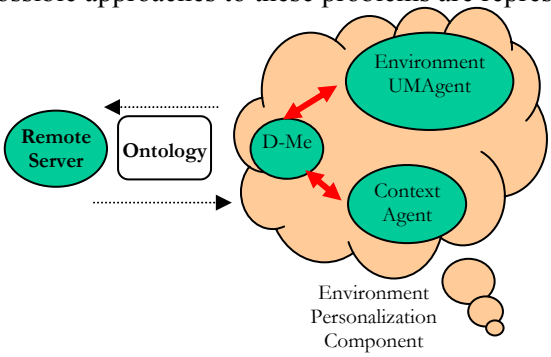

Fig.2 D-Me Personalization Component

All these approaches present advantages and disadvantages. In traditional client-server information systems, the most frequent design choice is to store the User Model on the server side, by enabling the user to access her/his model after having been recognized by the system. In the distributed solution, user information are stored in different servers, reducing in this way the computational load but presenting problems of redundancy/incompleteness of user information and consistency.

The mobile approach consist of a user which "brings" always with her/himself the user model, for instance on an handheld device, and, when the interaction with an environment starts, her/his profile is passed to the environment user modeling component. This approach seems to be very promising since it presents several advantages such as: the continuous availability of user information, wireless data communication, absence of information redundancy and easy management of consistency. However, we cannot assume that the user will have with her/himself an handheld device and this type of device still presents hardware-related limits (capacity, computational speed, battery,…).
In the D-Me architecture we adopt a distributed-mobile solution to the personalization (see Fig. 2).
On one hand, **D-Me** knows its user and how to perform delegated tasks. In order to accomplish its main desires it needs to reason on how to adapt tasks execution to "user in context" features, which user features have(are allowed) to be passed to the environment in order to get personalized service execution and how to presents results taking accordingly.
At this aim, it manages the user MUP, that could be stored on a capable user device or, if the interaction occurs through not powerful enough devices (i.e. a smart card), on a Remote Server trusted by the user. Information contained in the MUP together with the context situation and the task list, deriving by accomplishing the relative D-Me desires, may be used by D-Me to decide how to adapt its behaviour and which data to pass to the environment in order to get personalized service execution.
According to the previously mentioned scenario example, Fig.3 shows the MUP of a female second-year student. The XML structure reflects the user profile ontology used in D-Me and includes four main sections: **IDENTITY** (with identification data such as the user name, sex, id, password, and email), **MIND** (background knowledge, interests and know-how), **BODY** (disabilities or preferences in using a body part during interaction) and **PERSONALITY** (personality traits and habits). Every slot in the MUP can be protected by giving a 'scope' validity to the corresponding XML tag and can be made 'not public' by setting the 'publicly' attribute to 'false'.

```
IDENTITY   <UserIdentify slotName="userIdentity" login="lionettipam" password="miky75"
           publicly="false"/>
           <classIdentity slotName="Identity" surname="lionetti" telephone="332232"
           email=""
           name="pam" job="" born="21/7/77" sex="F" courseYear="II." resident="no"
           publicly="true"/>
           <interest slotName="Interest">
MIND       <UserConcept slotName="music" confidence="high" topic="pop"
           publicly="true" action="" target=""><EnvScope
           nameScope="All"/></UserConcept>
           <UserConcept slotName="study" confidence="high" topic="web programming"
           publicly="true" action="" target=""><EnvScope
           nameScope="DIB"/></UserConcept>
                    ---
           </Interest>
BODY       <Body slotName="Body"> <classBody use="true" part="Eye" publicly="true"/>
                    <classBody use="true" part="Hear" publicly="false"/>
           </Body>
           <Personality slotName="Personality">
PERSONALITY      <UserConcept confidence="high" topic="sociality" publicly="true"
                 slotName="userConcept" action="" target=""><EnvScope
                 nameScope="FreeTime" /></UserConcept>
....       </Personality>  ....
```

Fig3. An example of XML file representing a MUP

For instance, in Fig3. the student interest towards web programming can be shared with other agents only in the 'DIB', while her interest in pop music is always public. In the considered example, when the user is in the DIB, her identification data, interest in web programming and pop music will be considered, together with the preferred interaction modalities (visual interaction).

In the present prototype, data in the MUP are collected in the following three ways: i) the user can input information through a graphical interface, ii) other information (i.e. temporary interests) can be derived when the user insert a task in the To-Do-List (i.e. the second task in our example would generate an interest in "web programming"), iii) information inferred by the environment.
On the other hand, the **environment** can adapt service execution by adopting its own modelling strategy depending, for instance, on the nature of the considered environment. A commercial center could use a data mining approach to classify user shopping habits and adapt suggestions and advertisements, while another environment, i.e. a University Department, could use a different approach, for instance based on logical reasoning. In this case, only the data that are relevant to personalization of a particular service are passed to the environment UserModelingAgent (UMAgent), which starts the modeling process.
When interaction ends, the environment sends back to D-Me the inferred data that can be inspected, after a notification of D-Me, by the user accepting or refusing them.
As far as consistency is concerned, our approach gives to the D-Me agent, and then to the user, the task of keeping consistency between the MUP data and what has been inferred by an environment. We did not face yet the multi-environment consistency. However, since we provided the D-Me model with an infrastructure able to support federation of environments, at the moment we are considering how they could exchange information about the user. As for the underlying architecture, agents that interact to accomplish the user modeling task, communicates using ACL and sharing ontology. This enables us to overcome problems due to agents using different representations for user profiles [15].

Both entities, D-Me Agents and the Environment, need to sense and elaborate context information. In our approach, **Context** is grounded on the concept of "user task executable in an environment". Therefore, given a task in the user to-do-list, once the user has been classified according to the strategy of the UM component, its execution and results can be influenced by the context in which the interaction occurs and, in particular, by:

- ***static environment features***: an environment is made 'active' by giving to agents which populate it the capability to understand its features: that is, by modeling it. In particular an environment is described by its *scope*, *type*, *physical features*, and corresponding *map*. In My-DIB, we identified the following scopes: learning, social relations, leisure, food-services and administration. Every scope identifies a set of specialized services. For instance "learning" includes the following services: register for an exam, get information on how to prepare it, collect learning material and so on. The type of the environment refers to its social nature: public or private. Physical features includes a description of objects relevant for interacting with that environment. For instance, if there is a PC or an interactive totem that can be used to communicate results. Therefore, all the tasks in the To-Do-List that are enabled in the library and have 'learning' as a scope will be proactively activated when the user is in it or passes nearby according to their priority.
- ***dynamic environment features***: for instance noise and noise and light level;
- ***dynamic user features***, that identify the physical and social surroundings of the user(emotional state, location, activity the user is performing, time, ...);
- ***device*** employed and its state at the considered time (battery, connection, and so on.).

At the present stage of the prototype, we do not work on hardware sensors. They will be realized in the next stage. At the moment we simulate their values through an interface that sends the corresponding values to dedicated **Sensors Agents**, which communicate relevant changes to the **Context Agent** that knows the global context situation at the considered time.
In the considered example, the Sensor Agent controlling the user location detects the user presence in the DIB and in particular her relative position to key places such as the library. The Sensor Agent controlling the device detects that the user has got a PDA. The context situation relevant at time *ti* is represented in an XML structure compliant to the context ontology.

2.2. Context-Aware Task Models

In the model described above, adaptation to "user in context" features can be applied at different levels: by triggering user tasks, by asking specialized services in the environment to execute these tasks and by exchanging messages with the user: for instance, remind messages or information presentation. D-Me may

execute tasks in the To-Do-List that are enabled in the given context. To model the 'task-user-context' relation, we employed an extension of Petri Nets which was developed by our research group in a previous project [8].
For instance, in the previously described situation the first task in the To-Do-List corresponds to the D-Me goal *Remind(U, Do(Task, env, Cti))*, where *U* denotes relevant user features, *Task* denotes the task in the To-Do-List, *env* denotes the environment and *Cti* the context at time *ti*. In this case, when the user comes close to the library, D-Me generates a reminder for giving back the book that is presented appropriately by the Interface Agent. The second task in the To-Do-List activates another D-Me goal *Search(D-Me, News(U, Topic, env, Cti))*. In the current context, for achieving this goal, D-Me asks the library Service Agent whether Web Programming books are available and displays information according to student ratings.
Fig.4a shows the XDM model for the first task. In this model, the token (double coloured circle) contains information about the user and the context, detected at time *ti* relevant for executing it. The net represents the way in which D-Me executes the task according to this situation. Each place *pi* (circle) in the net represents the state of the system, each *u-transition* (square-box) represents a user action (both in the digital or real world), each *d-transition* (hexagon-box) represents a D-Me Agent action. This action can be of several types: i.e. detection of some context condition, to-do-list check, activation of new tasks corresponding to another net.

In particular, each place contains the following information: **user** and **context** features that have to match with the **token features** in order to move it in that place; **interaction state** (IS) that is activated if the token matches the user and context features. The interaction states represents a communication action of D-Me. The token content is dynamic since it is modified after a detection action. Double circled places represent a nested call to another net if the place conditions are satisfied. End places determines the exit from the net.
As for every model based on Petri nets, a transition is enabled and can fire if all the places in input to it contain a token and the condition attached to that transition are satisfied.
For instance, the net in Fig 4a. represents the context-aware model of a Remind Task (return the HTML book to the library) with low priority. Since it has a low priority we decided to trigger it when the user is near or inside the Library.
In this case the token features matches this constraints. At this point, after getting the remind the user may *return the book* (U1), *accept* the remind without returning the book (U2), *ignore* it (U3). D-Me is detecting relevant context features (S1). In this case it will detect that there is another D-Me task in the To-Do-List triggered by the current context: *Search(D-Me, News(U, Topic, env, Cti)*. The execution model of this task is another net (Fig. 4b) nested in place C4. C4 is accessible only if there are executable tasks as stated in its condition.

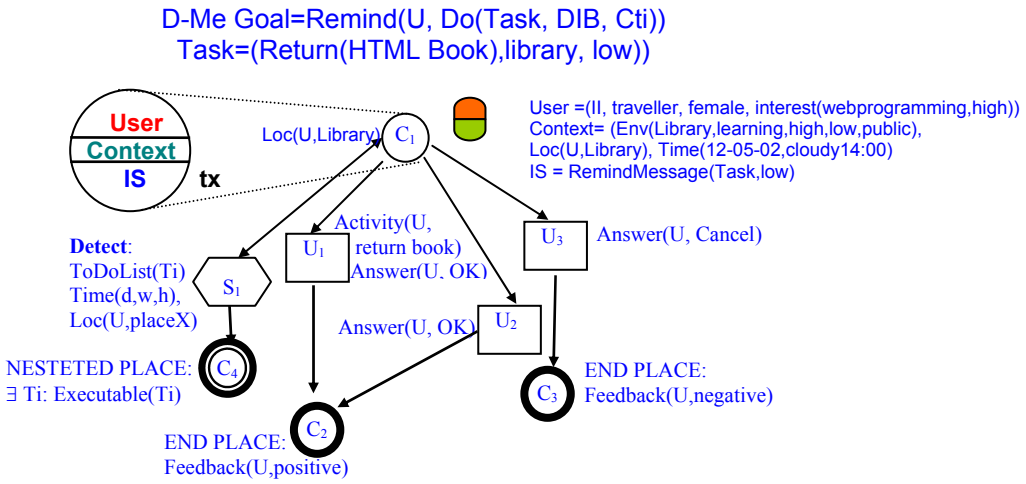

Fig. 4a. XDM model describing the a remind task

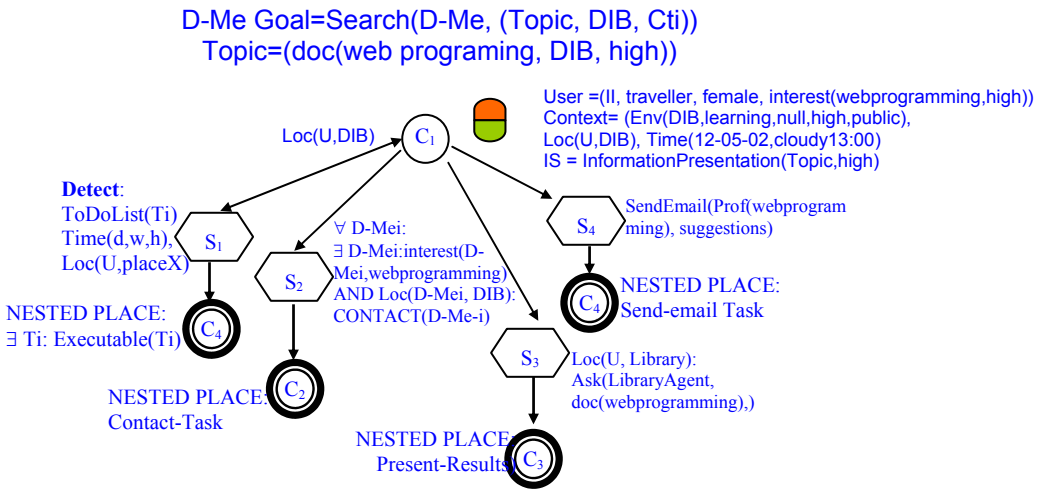

Fig. 4b. XDM model describing the a Search task

In particular, let's suppose that there are no other users in the DIB interested in Web Programming in that moment, only the S3 d-transition in Fig. 4b will be triggered. This will correspond to the D-Me action of asking to the LibraryAgent the execution of the requested search according to user preferences. In this case, relevant data in the MUP will be passed to My-DIB UMAgent that will categorize the user as a beginner in that topic but with some prerequisites (she is a second year student then she knows about programming, algorithm, and so on.). The Library Agent will take these information into account and will filter results accordingly.
Results will be then displayed appropriately by the D-Me Agent according to user preferences and context conditions. In particular this is the task of the D-Me Interface Agent as we will see in the next Section.

2.3 D-Me Interface Agent

This agent is an extension of the main D-Me Agent. It has the role of interacting with the user for communicating results of tasks or for asking information/confirmations required for task execution.

In My-DIB application, we consider the following families of communication tasks:

- ***request for input***. If, for instance, the to-do-list includes the task: "Sign for an exam next week" and the user is at work, D-Me will ask information about "which exam " and "at what time" the user wants to do it.
- ***information provision***: Information may be presented when explicitly requested by the user or proactively prompted by D-Me because related to the current user task. As we anticipated, in the case of the second task in our example, D-Me will display information about Web programming books present in the Libray according to student rating.
- ***request for confirmation***. Let's take as an example the previous task, let's suppose that one of its steps involves contacting by email the course professor for indication about suggested material to prepare the exam. Then, the D-Me agent will ask the user for confirmation before sending the email .
- ***notification messages***. Proactive task execution is notified by D-Me, for instance, in the previous case, if the agent has the autonomy to send emails it will not ask for permission and will just notify it.
- ***remind messages***. This is the typical message generated for the first task in our example.

User and context related factors are taken into account in generating the communication about a task in the following way:

1. **user preferences and features**: results of information provision tasks are filtered, ordered and presented according to what has been inferred about the user starting from her profile data (interest, know-about, know-how). Possible user disabilities are taken into account for media selection.

2. **activity**: this influences information presentation as follows. If the user is doing something with a higher priority respect to the one of the communication task, then the message is postponed until the current activity ends. If the communication regards the current activity, media used in the message take into account

the available body parts. Therefore, a voice input is preferable to a textual when, for instance, the user is running with her/his PDA asking for information about the next train to catch.

3. **location** of the user in the environment: texts, images and other media may be selected according to the type of environment (public vs. private, noisy vs. silent, dark vs. lightened, etc.) in which the users are and, more precisely, to their relative position to relevant objects in the environment.

4. **emotional state**: factors concerning the emotional state influence the level of detail in information presentation (short messages are preferred in stressing situation), the intrusiveness (bips and low priority messages are avoided when the user is already nervous), and the message content. For instance: if a user requests information in conditions of emergency, the agent will have to avoid engendering panic, by using reassuring expressions or voice timbre [5].

5. **device**: the display capacity affects the way information is selected, structured and rendered. For instance, natural language texts may be more or less verbose, complex figures may be avoided or substituted with ad hoc parts or with written/spoken comments [4, 9].

To accomplish the communication task, the Interface agent applies the following strategy: starting from XML-annotated results of a Service Agent, decides how to render them at the surface level taking into account the rules described above encoded in XSL.

3. D-Me: an Autonomous Agent

As mentioned in the introduction, a D-Me Agent is exhibits an autonomous behavior when achieving its desires. In particular, in the context of ubiquitous computing, we recognized the need to model autonomy at different levels:

- **Execution Autonomy**: related to execution of actions (tasks, subtasks, request of services, and so on).
- **Communication Autonomy**: related to the level of intrusiveness in communicating to the user. Can the agent take the interaction initiative in every moment or there are constraints related to the user and the context? Then, it is necessary to determine how much a message can be intrusive in a certain context.
- **Personal Data Diffusion Autonomy**: it is related to the autonomy of performing tasks requesting the diffusion of personal data like those contained in the user profile.
- **Resource Autonomy**: the agent may use critical resources of the user in order to executed delegated tasks (i.e. credit card number, time to schedule appointments).

Each dimension has an associated value that vary from "null" to "high" in a 5 values scale.
The "null" value represents the absence of autonomy, the system has to execute what explicitly requested by the user. It cannot infer any information or decide to modify task execution without explicitly asking it to the user. The opposite value, "high", represents the maximum level of autonomy that gives to the agent the freedom to decide what to do always according to constraints imposed by the user (i.e. budget limits). The other values determines an incremental growing of the autonomy in making decisions and inferring information [10].
For instance, the remind task in Fig. 4a, requires only a sufficient level of communication autonomy: D-Me has to remind the user to return the book. Since, this task has a low priority, if D-Me has an high communication autonomy, then it will use all the available media to communicate with the user. Otherwise, it will not bother too much the user and will generate a non intrusive remind message (i.e. just 1 bip and a textual message).
For executing completely the second task D-Me needs both execution and communication autonomy. If we consider the S4 d-transition in the net in Fig. 4b, if D-Me has a low execution autonomy on that task family and medium-high communication autonomy, it will remind the user to send an email to the professor responsible for the “web programming” exam as soon as the user is using a device able to send emails. While, in case D-Me has got a medium execution autonomy it will compose the email and ask the user to look at it and confirm the execution of that task before sending it to the professor. In case of high autonomy it will send it without asking for confirmation but it will just notify the event to the user.
Autonomy rules are revised according to the type of feedback the user provide to the agent: positive feedback enforces the autonomy on that category of task , negative feedback reduce it.

4. Discussion and Future Work

Effective ubiquitous interaction requires, besides techniques for recognising 'user in context' features, a continuous modeling of both the user and the context. Therefore, ubiquitous computing systems should be designed so as to work in different situations that depend on several factors: presence of a network connection, characteristics of interaction devices, user location, activity, emotional state and so on. As suggested by Dey [6], delegating to an agent tasks execution and presentation of personalised results may be a solution for dealing with the complexity of this problem.

This work is our first step towards supporting personalised interaction between mobile users and a smart environment. Every user is represented by a D-Me Agent that, according to the content of her/his "To Do List", performs tasks on the user behalf by negotiating services with the smart environment.

Interaction between these two entities is personalised according to policies that are implemented in a distributed way. D-Me knows its delegating user (MUP + to-do-list) and the Environment knows how to personalise its services accordingly (UM and Context Agents).

Since the interaction happens through a personal agent, we started to consider the "delegation-autonomy" adjustment necessary for achieving cooperation between the user and his/her representative. However, more work in understanding how the user feedback influences the level of autonomy especially when this feedback is implicit (until now we considered only explicit feedback).

Another aspect that has been designed and implemented in this system, but it is still subject of validation, is the group modeling component that is part of the environment (GroupAgent). This modeling component will be used by the environment to select relevant information on public displays according to recognized group preferences.

What we need to do now is to work on the interaction in terms of user-agent dialog metaphors (individual communication) vs. Environment-user-groups interaction (public communication).

REFERENCES

[1] Ardissono L, A. Goy, G. Petrone, M. Segnan and P. Torasso. Ubiquitous user assistance in a tourist information server. Lecture Notes in Computer Science n. 2347, 2nd Int. Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH2002), Malaga, pp. 14-23, Springer Verlag 2002.

[2] Byun H.E. & Cheverst K. User Models and Context-Awareness to Support Personal Daily Activities. Workshop on User Modelling for Context-Aware Applications. 2002.

[3] Chen G. and Kotz. D. A Survey of Context-Aware Mobile Computing Research. Technical Report TR http://citeseer.nj.nec.com/chen00survey.html.

[4] De Carolis B, de Rosis F, Pizzutilo S.Generating user-adapted hypermedia from discourse plans.AI*IA 97:Advances in Artificial Intelligence, LNAI M. Lenzerini (eds),Springer, 334-345, 1997.

[5] De Carolis B., de Rosis F. and Pizzutilo S. Adapting Information Presentation to the "User in Context". IJCAI 2001 Workshop on AI in Mobile Systems, Seattle, 2001.

[6] Dey A. K.Understanding and Using Context. . Personal and Ubiquitous Computing 5 (2001) 1, 4-7.

[7] Dey A.K. and G.D. Abowd: CyberMinder: A Context -Aware System for Supporting Reminders. Proc. Symposium on Handheld and Ubiquitous Computing, Bristol. (2000).

[8] de Rosis F., Pizzutilo S., and De Carolis B.. Formal description and evaluation of user adapted interfaces. International Journal of Human-Computer Studies, 49,1998.

[9] G. De Salve, B. De Carolis, F. de Rosis, C. Andreoli, M.L. De Cicco. Image Descriptions from annotated knowledge sources. IMPACTS in NLG, Dagsthul, July 25-28, 2000.

[10] Rino Falcone, Cristiano Castelfranchi: Tuning the Collaboration Level with Autonomous Agents: A Principled Theory. AI*IA 2001: 212-224.

[11] http://www.fipa.org.

[12] http://www.ai.sri.com/~cheyer/papers/aai/oaa.html

[13] http://www.protege.stanford.edu

[14] http://sharon.cselt.it/projects/jade/

[15] Heckmann, D. Ubiquitous User Modeling for Situated Interaction. 8th International Conference on User Modeling 2001, Proceedings. LNCS 2109 Springer. pp. 280-282

[16] Jameson A. Modeling Both the Context and the User. Personal and Ubiquitous Computing. Vol 5. Nr 1. Pp 29-33. 2001.

[17] Kobsa A., Generic User Modeling Systems. UMUAI vol. II nos.1-2 pp.49-63. Kluwer Academic Publisher. 2001.

[18] Maes, P. "Agents that Reduce Work and Information Overload," Communications of the ACM, Vol. 37#7, ACM Press, 1994.

[19] McCarthy, J. F. Active Environments: Sensing and Responding to Groups of People. Journal of Personal and Ubiquitous Computing, Vol. 5, No. 1, 2001

[20] Lieberman H. and Selker T. Out of Context: Computer Systems That Adapt To, and Learn From, Context. IBM Systems Journal, Vol 39, Nos 3&4, pp. 617-631, 2000.

[21] A. S. Rao and M. P. Georgeff, BDI-agents: from theory to practice, in "Proceedings of the First Intl. Conference on Multiagent Systems", San Francisco, 1995.

Speech as a Source for Ubiquitous User Modeling

Christian Müller and **Frank Wittig**
Department of Computer Science
Saarland University
{cmueller, wittig}@cs.uni-sb.de

Abstract

In this paper, we present an approach on how to use speech as a source for user modeling in a mobile and ubiquitous context. In particular, we exploit different abstraction levels of speech features to estimate the user's age and gender. To solve the classification task, we compared several well known machine learning techniques such as artificial neural networks and support vector machines. The results of our study imply that one can indeed successfully extract higher level information from the raw speech data. We show how this approach is integrated into a generic resource adaptive system architecture. One particular instance of this system is an implementation of a mobile pedestrian navigation system.

1 Introduction

In ubiquitous computing, speech plays an important role as an interaction modality. Application scenarios like mobile navigation systems, shopping, tourist or museum guides, imply hands-free eyes-free situations, where the users can interact with the system by speech only. Therefore, we consider speech as an important and rich source for ubiquitous user modeling. Speech contains information about the speaker: Hearing someone's voice, we can in most cases recognize the gender of the speaker, estimate the age, and maybe even get an idea of what mood the speaker is in with regard to stress or emotions.

The question of how this information can be made available for user modeling is currently addressed in the project M3I[1]. M3I is part of the BMB+F[2] funded project COLLATE[3] at the Saarland University and the DFKI[4]. The goal of the M3I project is to develop a framework for resource adaptive multi-modal dialog with mobile devices.

[1]A Mobile, Multi-modal, and Modular Interface

[2]Bundesministerium für Bildung und Forschung (Federal Ministry of Education and Research)

[3]Computational Linguistics and Language Technology for Real Life Applications

[4]Deutsches Forschungszentrum für Künstliche Intelligenz (German Research Center for Artifical Intelligence)

It consists of a central server component with which mobile devices can communicate via wireless network connections. While the mobile devices function on a stand-alone basis, the availability of the server improves the coverage and quality of the services. For example, the speech recognition that is implemented on the mobile device is limited due to lack of computing power and working memory. When connected to the server, the speech can be processed in parallel on the server much faster and with a larger vocabulary. Besides this, the server provides additional services like topic detection. That means that the server possesses a speech recognition module with a general language model that recognizes the domain to which the utterance of the user belongs. The integrated speech recognition module uses this information to load specific language models for this domain. The M3I framework is designed to implemented new components on the server side first. In this manner, the approach can be tested and improved easier before a slim embedded version is implemented. An example of such a component is the speech-based user modeling module that is described in this paper.

2 Features of Speech That Are Relevant for User Modeling

Regarding speech as a source for ubiquitous user modeling, the relevant features can be divided into three different levels of abstraction (see figure 1).

On the lowest level, there are *acoustic* features that are related to the signal's power and frequency and their changes over time. An example of such a feature is the jitter value that describes frequency variations between voiced periods of speech. Because they are based on physical properties, acoustic features are relatively easy to extract from the signal and independent from the language. They can be extracted before the actual speech recognition process is done. On the other hand, those features are sensitive to changes in the acoustic environment and the recording quality.

On the next level, there are *prosodic* features. Prosody refers to all aspects of sound above the level of segmental sounds, like intonation, stress and rhythm. Speech rate and pauses can also be assigned to this group. In most cases, prosodic features cannot be immediately derived from the physical properties. The extraction is therefore more expensive. In the case of speech rate and pauses they also have to

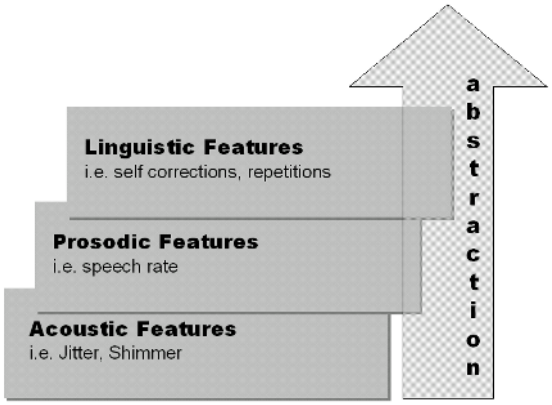

Figure 1: Three levels of abstraction of speech features

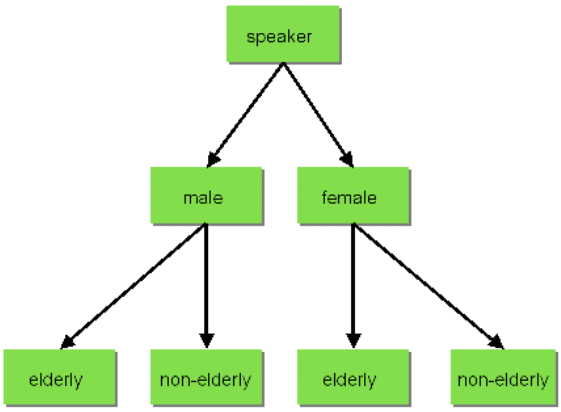

Figure 2: Classification hierarchy

be compared to a baseline, either of the individual speaker or of a group of speakers. Still, the extraction can be performed without understanding the content of the utterance.

This is no longer the case for *linguistic* features. Those features refer to the syntactical structure of the utterance, the number and category of the words, or even to their semantic content. To extract these features, natural language processing has to be done first.

Müller, Großmann-Hutter, Jameson, Rummer, and Wittig (2001) describe a study where prosodic and linguistic features were used to recognize the user's cognitive load and time pressure. The features were called "symptoms of cognitive load and time pressure" and were extracted manually from the speech by fully transliterating the utterances and rating the quality of the content. Some of the prosodic features that were found to be relevant for this task are: articulation rate (the number of syllables articulated per second of speaking time), silent pauses, and filled pauses (e.g., "Uhh"). Besides this, the following linguistic features were considered: (a) disfluencies (the logical disjunction of several binary variables, each of which indexes one feature of speech that involves its formal quality: self-corrections involving either syntax or content; false starts; or interrupting speech in the middle of a sentence or a word) and (b) content quality (the average quality assigned to the utterance).

The results were used for learning a Bayesian network (Pearl, 1988) that reflects the causal dependencies between the symptoms and the cognitive load and time pressure of the user. Müller et al. (2001) showed that this network can be successfully used for this particular classification task.

In the remainder of this paper we present an approach on how to extract acoustic and prosodic features of the speech for the purpose of recognizing the gender and the age of a user. In section 3 we provide a motivation, why the age of a user should be estimated by a system. In section 4, we present a case study on how to use machine learning techniques to induce classifiers for age and gender, and describe, how this approach is integrated into the above mentioned M3I architecture. In section 5 we briefly outline how the different abstract speech features introduced in figure 1 can be exploited within a single framework. We conclude in section 7 by directly referring to the relevant workshop questions.

3 The Elderly as a User Group

Elderly people are one of the last groups to benefit from access to computers. What makes technology difficult for elderly people to use is that they very often suffer from cognitive disabilities like age degenerative processes, motor impairments, short-term memory problems, and reduced visual and auditory capabilities (Jorge, 2001). These disabilities are often magnified by a person's unfamiliarity with the given technology and the different learning curves possessed by individuals. Making systems easier to use for elderly people raises two questions. First: What kind of adaptation should a system provide, when knowing that the current user belongs to the group of elderly users? And second: How can a system acquire this information? Müller and Wasinger (2002) address the first question by the example of a mobile pedestrian navigation system with a multi-modal dialog component. They suggest among other things that the speech output should be slower and the GUI should be clearer in that the toolbars, buttons, maps and text be displayed in a larger format. In this paper, we focus on the second question: How an appropriate user model can be obtained automatically on the basis of speech.

4 Case Study: Using Machine Learning to Induce Classifiers for Age and Gender on the Basis of Acoustic Features

In the following, we will present an initial exploratory study regarding the classification of users on the basis of low-level acoustic features according to their gender and age. Within a comparison of the most commonly used machine learning (ML) approaches, we aimed to find out whether it is possible at all to identify a user's age and/or gender with ML methods.

The voices of men and women age differently (Linville, 2001). Therefore it is reasonable to first try to determine the gender of the user, before the age is estimated. Figure 2 depicts the corresponding classification hierarchy.

By reviewing the literature, we identified *jitter* and *shimmer* as appropriate feature to determine the gender and the

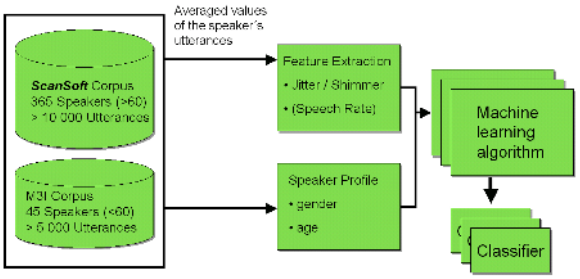

Figure 3: Age estimation procedure

age of the user (Linville, 2001; Schötz, 2001; Minematsu, Sekiguchi, & Hirose, 2002). Both features belong to the group of acoustic features according to the classification that was introduced in section 1. Besides this, the prosodic feature speech rate is also a candidate for age estimation, but has not yet been taken into consideration.

Jitter is defined as the maximum perturbation of fundamental frequency (F_0). Jitter values are expressed as a percentage of the duration of the pitch period. Large values for jitter variation are known to be encountered in pathological (and old) voices. Jitter in normal voices is generally less than one percent of the pitch period. Shimmer represents the maximum variation in peak amplitudes of successive pitch periods. Large values for shimmer variation are known to be encountered in pathological (and old) voices. Shimmer in normal voices is generally less than about 0.7db (see (Baken & Orlikoff, 2000));

Figure 3 depicts our approach. We analyzed a corpus with speech from elderly people that was provided by SCANSOFT[5] for this purpose. This corpus contained more than ten thousand utterances from 347 different speakers with an age of over 60 years. A second corpus that was collected within the M3I project contained about five thousand utterances from 46 speakers under 60 years. Table 1 summarizes both corpuses including the number of female vs. male speakers. We implemented feature extractors for jitter and shimmer using the open source phonetic analyzing tool PRAAT.[6]

We used five different jitter and three different shimmer algorithms that are provided by PRAAT. The main jitter algorithms are: *Jitter Ratio* (JR) , *Period Variability Index* (PVI), and *Relative Average Perturbation* (RAP), that are well known from the literature ((Baken & Orlikoff, 2000)), as well as the standard PRAAT jitter algorithm that is similar to RAP. The major differences are the following: JR determines cycle-to-cycle variability whereas PVI calculates a value that is akin to the standard derivation of a period. RAP compares the average of three cycles to a given period. In this vein, the effects of long term F_0 changes, such as slowly rising or falling pitch, are reduced. The differences between the shimmer algorithms are similar to the differences between the jitter algorithms. The *Amplitude Perturbation Quotient* for example attempts to desensitize long-term amplitude changes like RAP does for frequency variations. APQ uses eleven point averaging (average of eleven cycles). For a detailed description of jitter and shimmer algorithms, we refer to (Baken & Orlikoff, 2000).

All together, we received 8 features that can be used for classification. For the following initial study, the average values per person were used. In a future phase of the project, we plan to apply methods that continuously update the system's estimate of age and gender as more and more speech samples of the current user become observed. We are currently collecting more speech samples in order to work with a more balanced set, i.e. containing a larger number of non-elderly persons. Nevertheless, for an initial test whether the automatic classification is possible at all, the present data suffices, although we have to keep the uneven distribution of our samples in mind when discussing the results.

non-elderly	**elderly**
46	347

female	**male**
162	231

Table 1: Number of (non-)elderly and female vs. male persons in the data set

We performed for each learning/classification method that we applied a ten-fold cross-validation procedure. In our study, we considered the following methods (for the informed reader we list the key parameters in parentheses, if any): C4.5 decision tree induction (DT), artificial neural networks (ANN, learning rate 0.15, momentum 0.2, 500 iterations), k-nearest neighbors (kNN, k=5, simple distance weighting), naive Bayes (NB) and support vector machines (SVM, C=20, polynomial kernel with degree 4). Particularly, we used the implementations of the WEKA collection of machine learning tools (Witten & Frank, 1999).

Table 2 shows the results with regard to the predictive accuracy. As a baseline (BL), we included the results for a simple classifier that always predicts the more frequently occurring class, i.e. elderly (88%) and male (59%) samples, respectively. This enables us to interpret the results in a more adequate manner instead of simply looking at the raw percentages that may lead to wrong conclusions.

	C4.5	**ANN**	**kNN**	**NB**	**SVM**	**BL**
gender	69.10	81.73	76.41	67.26	70.43	58.78
age	92.41	96.75	95.76	91.25	96.45	88.30

Table 2: Results: prediction accuracy (percentages)

Overall the different results show that it is indeed possible to create classifiers that are able to successfully predict age and gender on the basis of low-level acoustic features of the user's speech. Each method performs significantly better than the baselines 58.78 and 88.3 (two-tailed t-test, $p<0.01$). Artificial neural networks perform best in our study. This is a reasonable result since this method is known to be successfully applied frequently in such situations where raw sensor data has to be exploited. Note, that naive Bayes is in both cases—the prediction of age as well as gender—the alternative that

[5] www.scansoft.com

[6] www.praat.org

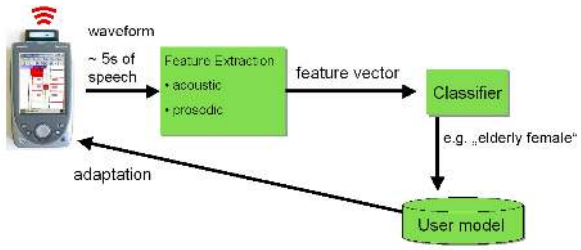

Figure 4: Integration into the M3I architecture

performs worst. This is most likely due to the fact that our data violates the basic assumption underlying the naive Bayes classifier: the independence of the feature values given the class value. Those 8 features used in our experimental setup are obviously not independent of each other. There are subsets that reflect mainly the same acoustic features of speech, i.e. variations of jitter and shimmer.

To get a better understanding of the performances of the classifiers with regard to our unbalanced data set, we present the true negative and positive rates in Table 3, respectively, i.e., the rates of correct predictions for the two separate classes. Particularly, we present the results for the artifical neural network.

non-elderly	elderly	female	male
82.6	98.3	69.8	88.7

Table 3: Results: true positive rates

These results show that although our data is way from being evenly distributed, the classifiers are able to predict each class correctly with a rate higher than 70%.

Nevertheless, as already mentioned, it is of minor interest which particular instance of the different learning/classification algorithms is able to outperform the others, the main result of our exploratory study is that we can indeed learn successful classifiers and that it is therefore worth to follow this line of research more intensively.

Figure 4 shows in a simplified way, how the age estimation component was integrated into the above mentioned M3I architecture. It is currently implemented as a server side service. The speech of the user is recorded on the mobile device (PDA) and then streamed to the server over a wireless network connection. On the server side, the relevant features are extracted and the corresponding values are used to classify the speaker according to age (elderly / non-elderly) and gender (female / male). The information is written into the user model and serves as a basis for adaptation.

5 Towards an Integrative Approach for Exploiting a Variety of Features of the User's Speech

As discussed in the introduction of this paper, speech in general is an important and rich source of information for ubiquitous user modeling. Each one of the three levels of abstraction of speech features as shown in Figure 1 may make its

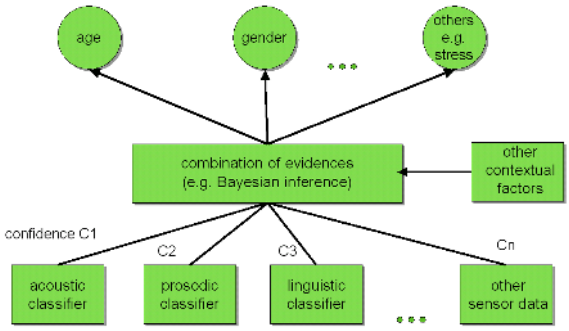

Figure 5: Integrative framework for exploiting a variety of features of the user's speech

own contribution in the context and user modeling process. Müller et al. (2001) have shown a system that is able to estimate a user's level of cognitive load and time pressure he/she is suffering from by interpreting high-level speech symptoms, e.g. self corrections, sentence breaks and so on. In this paper, we have described an initial study that strongly indicates that it is possible to successfully recognize some information about the user such as age and gender on the basis of low-level acoustic features of his/her speech. Related literature suggests that these features could also be used to reason about cognitive load and time pressure (Minematsu et al., 2002). In order to exploit a huge variety of available speech-related information for user modeling on the three different levels of abstraction, we briefly present an outline of an integrative approach along these lines.

Figure 5 represents the basic architecture. On the top, we have the variables of primary interest in our user/context model. These are connected to different classifiers that are used to interpret the different data streams on the different speech abstraction levels. To combine their results, each (single result of the) classifier comes along with a confidence value that measures the average success of the classifier. These confidence values could be estimated or computed on the basis of empirical studies as described in this paper or by Müller et al. (2001). If we interpret this architecture as the structure of a Bayesian network, these confidence values play the role of the conditional probabilities annotated at the links between the top row variables and the classifier variables, i.e., the probability that a particular classifier is able to correctly classify the speech symptoms in the situation under consideration. The implementation of this "meta-reasoning" scheme about the results of the different classifiers using BNs allows a flexible integration of other contextual factors or aspects of the domain that are relevant for the user model, such as input from bio sensors. The BN provides at any time up-to-date estimations in the form of probability distributions (conditioned on the available different pieces of information).

6 Coping with Mobile Speech and a Noisy Environment

When speech is recorded with mobile devices such as handheld PC, the quality of the signal is reduced due to the low

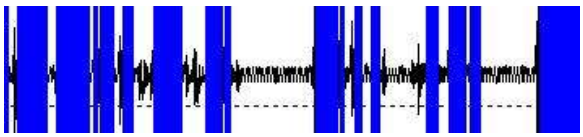

Figure 6: Voiced portions of the speech signal

microphone quality and environmental noise. This raises the question, whether the classifiers that were trained with clean data still perform well, when applied to mobile data.

First tests with a mobile device (HP Jornada) in fact showed, that the classification performance was far from the above mentioned cross validation results. Whereas gender was mostly recognized correct, young voices were often wrongly classified as elderly. However, we ascribe this fact partly to the unbalanced corpus (far more elderly voices) that may lead to a biased classification. In our approach, the effect of noise (environmental and microphone-induced) is reduced by the fact, that jitter and shimmer algorithms are based solely on voiced parts of the signal. As illustrated in figure 6, only a subset of the signal is treated (between 75 Hz and 600 Hz).

Nevertheless, we cannot exclude any impact of signal quality. Currently, we pursue two approaches to investigate this issue: (a) we implement filters that artifically reduce the quality of our training material. Thereby the characteristics of mobile recorded speech are simulated as close as possible. The performance of the resulting classifiers can then be compared with the one that were trained with unmodified data; (b) on a higher level we intend to incorporate a node NOISE into the above mentioned Bayesian network structure to model the impacts of noise explictly (e.g. reduce the confidence values of the classification when noise is likely present). Whereas (a) is under way, (b) requires a better understanding of the impacts of noise and will be focus of further research.

7 Conclusion with Regard to Workshop Questions and Current Work

In this paper we showed how to exploit raw speech data to gain higher level information about the user in a mobile context. In particular we introduced an approach for the estimation of age and gender using well known machine learning techniques. We classified the relevant speech features into three levels of abstraction each implying their own characteristics with regard to extraction costs and expressiveness.

We introduced the architecture of the M3I project, which copes with the limited resources of the mobile scenario by distributing services between mobiles devices and a server (see section 1). The age and gender estimation component that is described here was integrated into this architecture. A demonstration of the system can be given at the workshop.

Application scenarios within the M3I include a mobile pedestrian navigation system with a multi-modal interface. Such an application benefits from the advanced user modeling by (a) the facility adapting the interface with regard to the special needs of a particular user group (the elderly) and (b) the improved speech recognition quality using specific acoustic models.

Currently, one line of work is collecting more data to balance the corpus, investigating the impacts of noise as described in section 6, and the implementing extractors for prosodic features such as speech rate. Another line consists of the concrete realization of the above mentioned framework to integrate information of all three levels of speech features.

References

Baken, R., & Orlikoff, R. (2000). *Clinical measurement of speech and voice (2 nd edition).* San Diego: Singular publishing Group.

Jorge, J. A. (2001). Adaptive tools for the elderly: new devices to cope with age-induced cognitive disabilities. In *Proceedings of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing* (pp. 66–70). ACM Press.

Linville, S. E. (2001). *Vocal Aging.* San Diego, Ca: Singular.

Minematsu, N., Sekiguchi, M., & Hirose, K. (2002). A perceptual study of speaker age. In *Proceedings of the International Conference of Acoustics Speech and Siganl Processing* (pp. 123–140).

Müller, C., Großmann-Hutter, B., Jameson, A., Rummer, R., & Wittig, F. (2001). Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), *UM2001, User Modeling: Proceedings of the Eighth International Conference.* Berlin: Springer. (Available from http://dfki.de/∼jameson/abs/MuellerGJ+01.html)

Müller, C., & Wasinger, R. (2002). Adapting Multimodal Dialog for the Elderly. In *Proceedings of the ABIS-Workshop 2002 on Personalization for the Mobile World.*

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

Schötz, S. (2001). A perceptual study of speaker age. In A. Karsson & J. Van de Weijer (Eds.), *Proceedings of Fonetik 2001* (pp. 136–139). Lund Working Papers.

Witten, I. H., & Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations.* Morgan Kaufmann Publishers.