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.

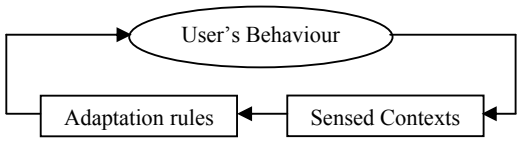

Figure 1. The relations between a user, the user's behaviour, and contexts.

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.

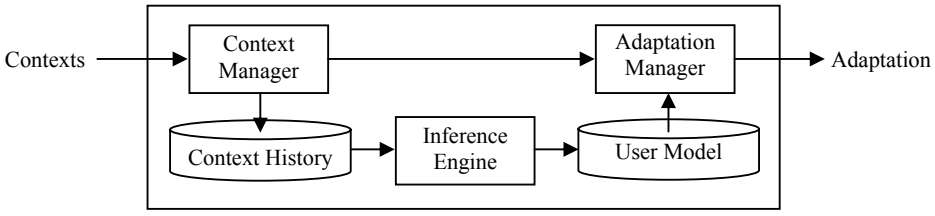

Figure 2. The design for providing dynamic adaptations.

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.

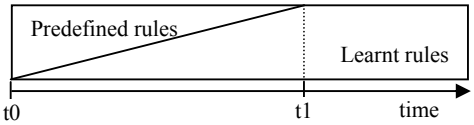

Figure 3. The phase of transition for the initiative of adaptations.

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.

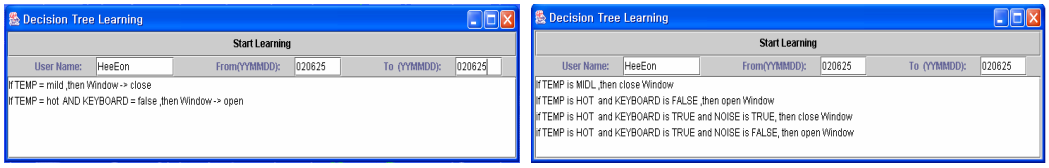

Figure 4. Learnt rule sets from different size of context history

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.

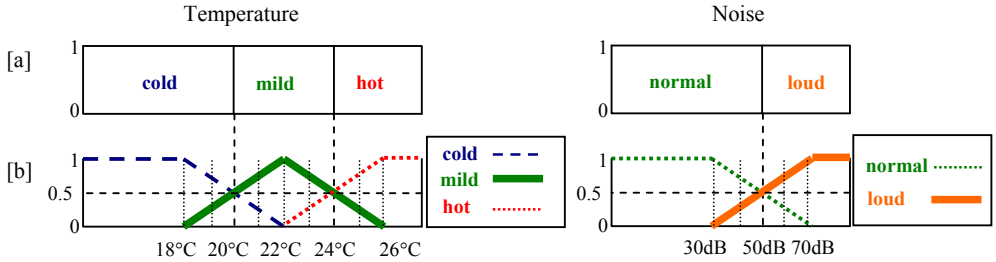

Figure 6. Conventional and fuzzy representation of temperature and noise ranges.

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.

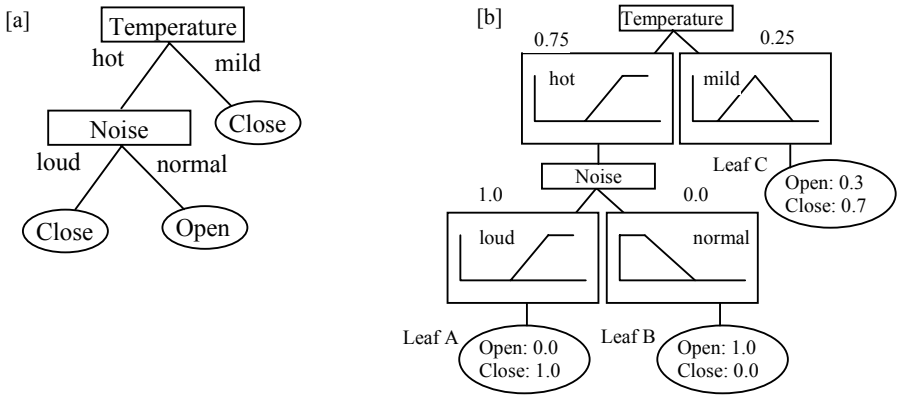

Figure 7. [a] a conventional decision tree and [b] a fuzzy decision tree
- given Temperature of 25 degrees and a Noise level of 70 decibels

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.

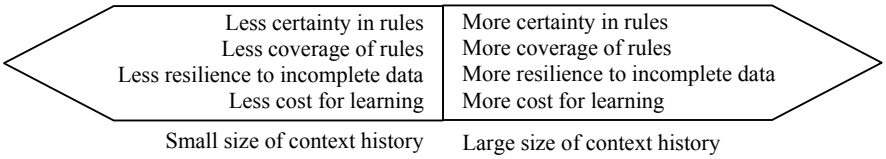

Figure 8. The characteristics of learning system based on the size of context history.

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.

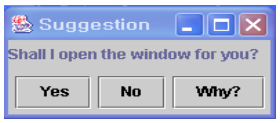

Figure 9a. A suggestion window.

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.