Implicit Adaptation of User Preferences in Pervasive Systems

Sarah McBurney, Elizabeth Papadopoulou, Nick Taylor, M. Howard Williams
School of Math & Computer Sciences
Heriot-Watt University, Riccarton,
Edinburgh, UK
{ceesmm1, ceeep1, nkt, mhw}@macs.hw.ac.uk

Abstract— **User preferences have an essential role to play in decision making in pervasive systems. However, building up and maintaining a set of user preferences for an individual user is a nontrivial exercise. Relying on the user to input preferences has been found not to work and the use of different forms of machine learning are being investigated. This paper is concerned with the problem of updating a set of preferences when a new aspect of an existing preference is discovered. A basic algorithm (with variants) is given for handling this situation. This has been developed for the Daidalos and Persist pervasive systems. Some research issues are also discussed.**

Keywords- user preferences; pervasive systems; machine learning

I. INTRODUCTION

Since the initial vision of ubiquitous environments was proposed in the early 1990s, many projects have attempted to achieve this vision using increasingly ambitious techniques. An innovation of initial projects such as Microsoft's Easy Living project [1] was context-awareness (which is now taken for granted in achieving pervasiveness). In this case much effort was focused on gathering information regarding the situation of users rather than the users themselves. As a consequence, environments adapted in exactly the same way for different users, regardless of individual needs or characteristics.

Projects such as IBM's Blue Space [2] and Intelligent Home [3] identified the benefits of personalization mechanisms and incorporated user preferences into system decision-making. This allows for adaptation specific to the needs of individual users. The preference set is created and maintained explicitly by the user through some GUI. Unfortunately the burden on the user to perform such information management tasks is great and hence users avoid it. This leads to sparse preference sets and basic personalization.

The need for more complete preference sets was soon realized. However, in addition to the problem of obtaining a more complete initial set of preferences, there is a continuing need for maintenance of the set of preferences. This is due to the fact that user needs will change with time for a variety of reasons, e.g. changes in lifestyle (moving house, starting a new job, going on holiday, etc.), the availability of new services or simply due to a 'change of mind'. Such changes will impact on the user's preference set requiring revisions to keep all preferences up to date. Ultimately the user should be given final control over their preference set but it was realized that, where possible, implicit techniques should be implemented to support the user and maintain the preference set on his/her behalf in an unobtrusive way. In such a system, preference adaptation will occur explicitly (via user input through some preference GUI) or implicitly (using machine learning).

For these reasons many projects include implicit learning techniques with monitoring and learning mechanisms to create and maintain the user's preference set unobtrusively on their behalf. The Adaptive House [4], MavHome [5] and GAIA [6] have full automation as their key goal. There is no support for explicit user input, all preference adaptation occurs implicitly. However, the offline nature of the learning algorithms employed could lead inevitably to frustrating situations where user behaviour has changed but the preferences do not reflect this. The Synapse project [7] attempts to remedy this problem by providing a distinction between preferences that have a high system confidence and preferences with a low system confidence. Several projects, such as Mobilife [8] and Spice [9], process information in real-time as well as processing stored history data in batch mode in order to provide a quick response to behaviour changes.

Allowing for explicit adaptation requires that preferences can be presented to the user in a human-readable format so that revisions can be made manually. In this case the solution lies in designing effective GUIs to facilitate this.

However, the use of implicit adaptation presents further challenges. For example, whenever new preferences are discovered they need to be merged with the existing preferences in a way that will hopefully make sense to the user if he/she views them. Conflicts may arise during the preference updating process and expecting the user to make the final decision is often not possible since the user may not be present to provide input (e.g. if the preference learning and updating cycle is running overnight). Therefore policies must be in place to indicate how implicit processes should overcome conflicts.

This paper describes a basic algorithm (with variants) for doing this.

Daidalos is a large European research project, a major aim of which is to develop a pervasive system [10], focussing especially on mobile users while Persist is another European project focussing on Personal Smart Spaces. The algorithm outlined here has been developed for use in both the Daidalos and Persist pervasive systems.

The next section provides a brief illustration of the problem. Section 3 presents the algorithm to handle the problem of preference merging. Section 4 discusses briefly how this is used in the Daidalos pervasive system. Section 5 discusses some research issues that still need to be addressed.

II. Illustration of the Problem

As we have noted, the main problem with user preferences is the difficulty in obtaining them and in keeping them up to date. One could ask the user to input a set of preferences and change them appropriately when required but this has been found not to work. One could provide a set of stereotypes to assist the user in setting them up but these will only approximate the user's real requirements and one still has the problem of dealing with new requirements or changes to the existing set with time. One could rely on machine learning techniques but these on their own are not sufficient.

The ideal solution lies in a combination of user input and machine learning. If one uses a rule based format that is easy for the user to understand and for machine learning techniques to generate then it is possible for the user to create preferences whenever he/she wishes or for machine learning techniques to create or amend the set of preferences at any stage. Equally the user can at any stage view the consolidated set of preferences and change them as needed.

Consider the following simple example. When our user, John, sets up his preferences initially, he creates a preference for "VoIP" (Voice over IP). He sets this up to select MSN Messenger when he is at home (as it contains his personal contacts) and Skype while at work (as it contains his business contacts). Suppose that he has a "User Agent" (or whatever one wants to call this software) running on his PDA through which he communicates with the system. This means that whenever he selects "VoIP" on his PDA at home the system will select and start MSN Messenger while at work it will select and start Skype, i.e.

```
IF location = home
THEN
voip = MSN Messenger
ELSEIF location = work
THEN
voip = skype
```

However, the machine learning system will monitor John's actions and may notice that when he is in his car he also selects Skype and when he is walking around in town he also selects MSN Messenger. In these cases it would want to create new sub-rules and preferably merge these with the existing rule to create a single composite rule for "VoIP".

```
IF location = home OR location = town
THEN
voip = MSN Messenger
ELSEIF location = work OR location = car
THEN
voip = Skype
```

Besides service selection, user preferences may be used to personalize the services themselves in various ways. For example, if John goes into a lecture room, he may switch his mobile phone to silent mode. If he does this several times the machine learning system will create a preference rule that sets the parameter for the mobile phone service automatically when this occurs. This situation was demonstrated in the Daidalos system. Similarly, telephone calls may be redirected depending on the caller and the user's context (again demonstrated in Daidalos). Once again this can be learnt and stored as a preference. And so on.

Here we are assuming that obtaining location information is no problem. This is one of the fundamental assumptions of most ubiquitous/pervasive systems. We are also assuming that location can be handled as a discrete variable with a small set of relevant values. This will be discussed further in section 5. If one includes time as a possible variable this too must be handled as a discrete variable. Another important variable is the current task, which is already discrete.

III. Algorithm for Merging

In order to learn from the user's actions, a record must be maintained of the history of these actions. In the case of user preferences, the actions indicate either user acceptance of the system decision based on the preferences or rejection of it (i.e. success/failure).

In our case a dual store approach is used [11] consisting of short-term (since the last data mining operation) and long-term (the complete set as far as possible) components. This section outlines the different situations that can arise when merging existing preferences from the user's profile with new preference information (obtained from mining the short-term component) and hence the basic algorithm.

The format used for user preferences in Daidalos is a simple or nested IF-THEN-ELSE rule. The condition part of the IF-THEN-ELSE is a Boolean expression consisting of context comparisons of the form *context attribute, relational operator, context value* joined with AND, OR and NOT. Daidalos context is as defined in [12, 13]. The THEN-part and ELSE-part can be another IF-THEN-ELSE construct, an assignment statement (attribute = value) or a constraint of the form

attribute relationalop value

where the attribute is the name of the parameter being personalized (e.g. volume) and is also equal to the preference name, and the value is a possible value of the personalizable parameter. When a preference is evaluated the result is referred to as the *preference outcome* and this is also a tuple of the form attribute, operator, value, where operator is either assignment or a relational operator.

In addition each preference outcome has associated with it a confidence level. This reflects the degree of confidence that the outcome is correct for the related context conditions at any point in time. For example, when the user creates a rule, the confidence levels of the outcomes are set to 1 but as it is updated, the levels may be any number >0 and <=1.

During the preference learning process, initial confidence levels are calculated for each learned preference outcome. They indicate how often a particular outcome has been observed with the related context condition in the user's behaviour history. Such information can help resolve conflicts that occur while updating the user's preference set with new learned preferences. Confidence levels also infiltrate through to personalization processes where they determine how and if a preference outcome should be applied.

When merging preferences, the basic rule that is adopted is to take the disjunction of the two. Boolean algebraic laws may be used to simplify rules when they become complex.

Consider now the different situations that can occur when merging existing preferences from the user's profile with new preference information created by the learning components. These form the basis for creating algorithms for merging preference conditions.

Situation 1
Preference Conditions: Identical
Preference Outcomes: Different

In this case, there is a preference stored in the user's profile that indicates that:
IF A= =a THEN X=x
while the new preference information indicates that:
IF A= =a THEN X=y

This represents a direct conflict between what was believed to be true and the new observation.
Solution: There are three ways to handle this situation:

a) Between the old and new preference, discard the preference with the lowest confidence level and decrease the confidence level of the preference with the highest confidence level using an appropriate algorithm. The algorithm should enforce the rule that the lower the confidence level of the weaker preference, the less the confidence level of the stronger preference should be reduced. e.g. if Preference P1 has confidence level of 0.65 and Preference P2 has a confidence level of 0.10 then P1 has a higher probability that it is correct than P2 so the confidence level of P1 should not be reduced too much. If Preference P2 has a confidence level of 0.60, this means that both preferences have roughly the same probability of being correct and thus the confidence level of the stronger preference should be decreased significantly. The weaker preference is discarded.

b) Instruct the learning component to extract all relevant actions from the long-term history and re-mine the data for this preference name, resolving the conflict.

c) Prompt the user to select which preference should remain and which one should be discarded or allow the user to edit the preference to explicitly state her wishes.

Situation 2
Preference Conditions: Identical
Preference Outcomes: Identical

In this case there is a preference stored in the user's profile which indicates that:
IF A= =a THEN X=x
New preference information indicates exactly the same:
IF A= =a THEN X=x

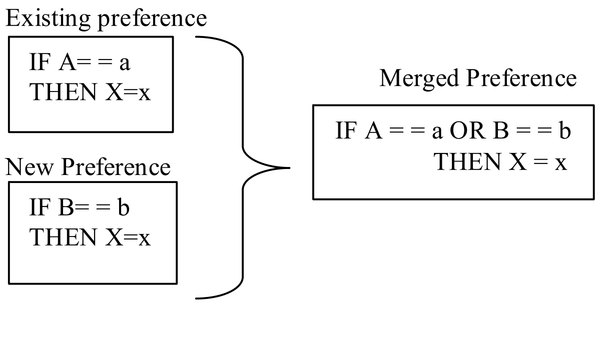
Solution: Since the information is the same, the confidence level of the existing preference is reinforced, thereby raising it. The formula used in situation 1 to determine the confidence level of the existing preference is used to recalculate it. In this case, the new confidence level will be higher because the number of times this preference has been inferred is higher than the previous time the level was calculated.

Situation 3
Preference Conditions: Different (variable)
Preference Outcomes: Identical

There are 4 different component situations here depending on the conditions:

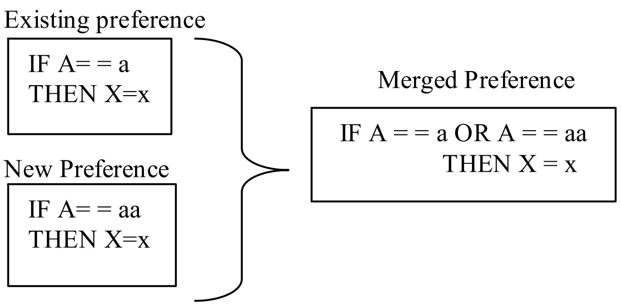
Situation 3.1
Context attributes: All different
Context attribute values: Not Applicable

Here the context attributes that make up the condition of the existing preference are all different from those of the new preference. Thus the two conditional parts are merged using an OR operation.



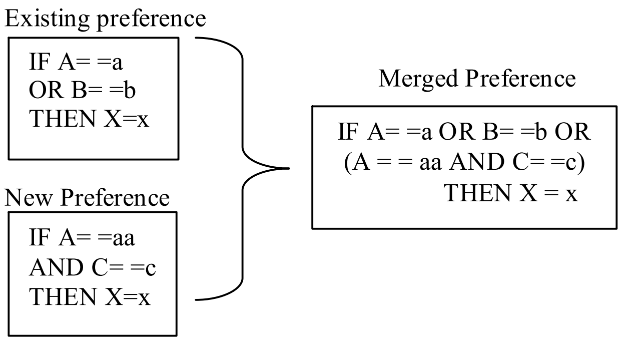
Situation 3.2
Context attributes: Identical
Context attribute values: Different

In this situation, all context attributes that make up the preference condition of the existing preference match those of the preference condition of the new preference. However, they each have **different values.** Once again the conditions can be joined by an OR operation, with no need to do anything else. If AND had been used, one would have to ensure that one did not end up with an expression of the form IF A= = a AND A = = b (always false). Therefore



Situation 3.3
Context attributes: Some different, some identical
Context attribute values: Different

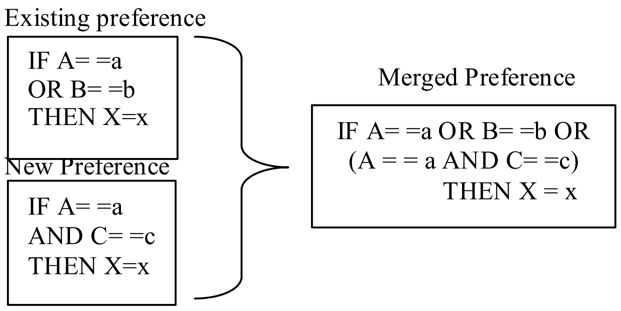
In this situation some context variables are different and some are the same but with different values. Once again the conditions are joined with an OR operation.



Existing preference
IF A= =a OR B= =b THEN X=x
New Preference
IF A= =aa AND C= =c THEN X=x
Merged Preference
IF A= =a OR B= =b OR (A = = aa AND C= =c) THEN X = x

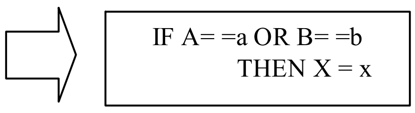
Situation 3.4
Context attributes: Some different, some same
Context attribute values: Identical

Some context attributes are different, the rest are the same with the same values. Again they are joined with an OR.



Existing preference
IF A= =a OR B= =b THEN X=x
New Preference
IF A= =a AND C= =c THEN X=x
Merged Preference
IF A= =a OR B= =b OR (A = = a AND C= =c) THEN X = x

Applying transformations this could be simplified to:

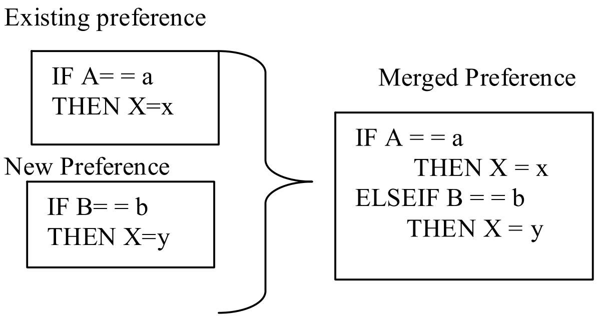


Situation 4
Preference Conditions: Different (variable)
Preference Outcome: Different

As in Situation 3, this situation also has subcategories:

Situation 4.1
Context attributes: All different
Context attribute values: Not Applicable

Here all context attributes in the conditions of both preferences are distinct. Create a nested IF-THEN-ELSE rule.



One problem here is the possibility that expressions <A = = a> and <B = = b> both evaluate to true and the order in which they exist in the IF-THEN-ELSE construct will define which outcome will be executed.

The solutions are:

a) Check the confidence levels of each IF-THEN branch. Set the condition with highest confidence level as the first condition to be evaluated and the other as an ELSE statement. This ensures that the condition with highest confidence is always executed when it is true.

For example, if A = = a has a lower confidence level than B = = b then the merged preference would be:

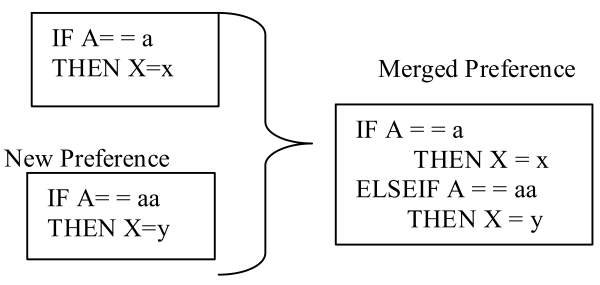
```
IF B = = b
        THEN X = y
ELSEIF A = = a
        THEN X = x
```

b) Instruct the learning component to re-mine the long-term history and replace the preference with the outcome of the mining algorithm.

<u>Situation 4.2</u>
Context attributes: All same
Context attribute values: All different

Here all the context attributes from the condition part of the existing preference are the same as those of the new preference but their values are all different. Create a nested IF-THEN-ELSE rule.

Existing preference

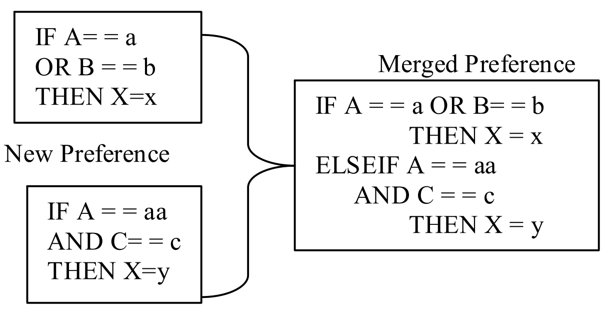


```
IF A= = a
THEN X=x

IF A= = aa
THEN X=y

IF A = = a
     THEN X = x
ELSEIF A = = aa
     THEN X = y
```

<u>Situation 4.3</u>
Context attributes: Some different, some same
Context attribute values: All values different

Some context attributes are different between the two preferences, some are the same but have different values. Again create a nested IF-THEN-ELSE rule.

Existing preference



```
IF A= = a
OR B = = b
THEN X=x

IF A = = aa
AND C= = c
THEN X=y

IF A = = a OR B= = b
        THEN X = x
ELSEIF A = = aa
    AND C = = c
        THEN X = y
```

One problem here is that there is a possibility that the expressions <A = = a OR B= = b> and <A = = aa AND C = = c> both evaluate to true and the order in which they exist in the IF-THEN-ELSE construct defines which outcome will be executed.
Possible solutions include:

a) Check the confidence levels of each IF-THEN branch. Set the condition with highest confidence level as the first condition to be evaluated and the other as an ELSE statement. This ensures that the condition with highest confidence is always executed when it is true.

For example, if <u>A = = aa and C = = c THEN X = x</u> has a higher confidence level than <u>A = = a OR B= = b THEN X = y</u> then the merged preference would be:

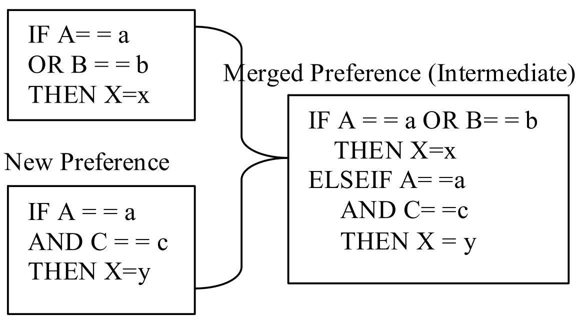
```
IF A = = aa and C = = c
    THEN X = x
    ELSEIF A = = a OR B= = b
    THEN X = y
```

b) Instruct the learning component to re-mine the long-term history and replace the preference by the outcome of the mining algorithm.

<u>Situation 4.4</u>
Context attributes: Some different, some same
Context attribute values: All values same

In this situation, some context attributes are different while some are the same with the same values.

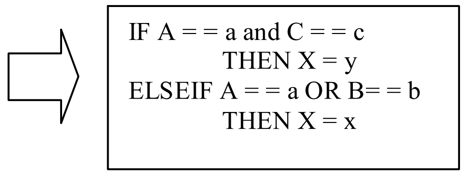
Existing preference



```
IF A= = a
OR B = = b
THEN X=x

IF A = = a
AND C = = c
THEN X=y

IF A = = a OR B= = b
   THEN X=x
ELSEIF A= =a
    AND C= =c
    THEN X = y
```

Suppose that the new preference has a higher confidence level than the existing one. Then the order of the conditions will be reversed, i.e.

Final Merged Preference



```
IF A = = a and C = = c
        THEN X = y
ELSEIF A = = a OR B= = b
        THEN X = x
```

IV. The Daidalos Pervasive System

Daidalos is a large European research project, whose overall aim is to create a pervasive environment for mobile users by integrating a range of heterogeneous networks and devices and creating a pervasive system on top of this. This will protect the user from the complexity of the underlying infrastructure while providing personalized and context aware services with minimal user intervention. The research is divided into two phases with slightly different objectives, spread over a five year period. The work is now in its final stages.

The pervasive system (or pervasive service platform) is based around the following six functions:

(1) Service Discovery and Selection.
(2) Service Composition.
(3) Session Management.
(4) Personalization.
(5) Context Management.
(6) Security and Privacy.

In the first phase capture of user preferences was done manually while in the second phase different learning techniques were included to support the build up and maintenance of the user profile. This necessitated the incorporation of some form of preference merging and for this an instance of the algorithm outlined in section 3 was used.

In our implementation one could have more than one preference rule associated with the same action (e.g. selecting a service or personalizing a particular parameter for a service). Each rule has a confidence level associated with it and when evaluating a preference the rule with the highest confidence level is selected. The reason for maintaining multiple rules is that these rules with lower confidence levels may become important as one discovers new preference information.

To handle the re-evaluation of confidence levels (Situation 1), the following general parameters were considered to be important:

a) The time that has elapsed since the last occasion when the preferences were updated (t_u).
b) The number of times the application of the outcome was reversed by the user (n_r).
c) The time that has elapsed since the last occasion when the application of the outcome was reversed by the user (t_r).
d) The lifespan of the preferences (t_l).

In addition the following factors are relevant to each specific preference rule:

(e) The confidence level of the existing preference (C_e),
(f) The number of times that it has been inferred (n_i),

From these factors the following are derived:

C_{eu}: updated confidence level of existing preference
C_{new} : confidence level of new preference.

Using the algorithm shown in Fig. 1, the confidence levels of the preferences are updated and then stored. During the evaluation of the preferences, the system uses the one with the highest confidence level ignoring the preferences with lower confidence levels.

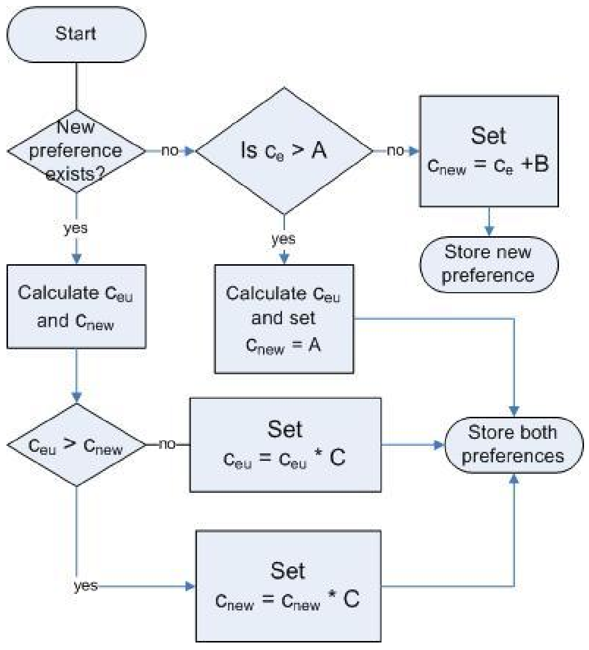


Figure 1. Algorithm used for confidence level update

In implementing the algorithm for preference merging itself, at every point where more than one option was available, the first option was selected.

Furthermore, for the constants in Fig. 1, the following values were used:

A = 0.25
B = 0.01
C = 0.20

Following a brief empirical evaluation of these choices for the solution which showed that they work adequately, this approach has been implemented and will be demonstrated shortly.

V. Some Research Issues

(1) One of the biggest research issues is the choice of machine learning technique to use. In both Daidalos and Persist we are experimenting with three different types of techniques – rule-based, Bayesian and neural net. This paper is concerned with the first of these. We are also using a combination of offline data mining and online incremental learning. However, there are many different techniques available and the question remains as to which is the best technique to use in which situations and how frequently it should be applied. Much work remains to be done here.

(2) When a new user preference is found, it must be due to one of the following:

(a) The user's behaviour has changed and we have discovered a pattern in the new behaviour (new).

(b) A new condition has arisen and we have discovered a pattern for this new condition (addition).

(c) A temporary change in the user's behaviour has occurred but should not result in a change to the preference (temporary).

(d) An old pattern has been re-discovered but the user's behaviour has changed and the preference has been updated to reflect the new behaviour (old).

Thus another major problem lies in distinguishing whether the newly identified rule is a permanent change in the user's preferences or a temporary one that should not result in an update to the user's preferences. To resolve this, the user should be involved whenever this is possible. However, with offline data mining this is not always possible.

(3) There is an infinite set of possible formulas for calculating and re-calculating confidence levels. For example, one might include the average time between inferences. Determining the ideal set of formulas is a subject of further research.

(4) Simplifying the conditions of rules for completely general formulae is NP-complete. However, for the types of rules we are generating some simplifications are possible and it is desirable to try to present to the user simplified rules wherever this can be done.

(5) As noted previously, we are not concerned here with the problems of context awareness and maintaining an accurate set of context values. We are assuming that the context system can obtain the user's location and provide it as a small set of strings with which the user is familiar. In the case of time, we are assuming that this can be divided into a small set of appropriate intervals in a similar fashion. Although these are important challenges in themselves, they are being addressed by others.

(6) Extending the ideas to the learning of preferences for groups of users presents a further challenge.

VI. CONCLUSION

The problem of building up and maintaining a set of user preferences for an individual user in a ubiquitous or pervasive environment is a nontrivial problem. And yet it is one of the most important issues that need to be solved if such systems are to be acceptable to the user.

In the Daidalos project a pervasive system is being developed in which different techniques are used to assist in this task.

This paper addresses one of the problems that need to be dealt with in building up user preferences – namely that of merging newly discovered preferences with existing ones. It presents a basic algorithm for solving this problem. This has been developed for use in the current implementation of the Daidalos prototype and will be further studied in the implementation of the Persist pervasive system.

ACKNOWLEDGMENT

This work was supported in part by the European Union as part of the Daidalos project under the Sixth Framework Programme and the PERSIST project under Framework 7. However, it should be noted that this paper expresses the authors' personal views, which are not necessarily those of the Daidalos consortium, and that apart from funding these two projects, the European Commission has no responsibility for the content of this paper.

REFERENCES

[1] B. Brummit, "Better Living Through Geometry", Journal on Ubiquitous Computing, pp 42-45, 2001.
[2] S. Yoshihama, P. Chou, and D. Wong, "Managing Behaviour of Intelligent Environments", Proc. PerCom '03, pp 330-337, 2003.
[3] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S.X.Q. Zhang, "The Intelligent Home Testbed", Proc. Anatomy Control Software Workshop, 1999, pp 291-298.
[4] M.C. Mozer, "Lessons from an Adaptive House", *Smart Environments: Technologies, protocols and applications*, 2004, pp 273-294.
[5] M.G. Youngblood, L.B. Holder and D.J. Cook, "Managing Adaptive Versatile Environments", Proc. PerCom *'05*, 2005, pp 351-361.
[6] B.D. Ziebart, D. Roth, R.H. Campbell, and A.K. Dey, "Learning Automation Policies for Pervasive Computing Environments", Proc. ICAC '05, 2005, pp 193-203.
[7] H. Si, Y. Kawahara, H. Morikawa and T. Aoyama, "A Stochastic Approach for Creating Context-Aware Services based on Context Histories in Smart Home", Proc. ECHISE *2005*, 2005, pp 37-41.
[8] C. Cordier, F. Carrez, H. Van Kranenberg, C. Licciardi, J. Van Der Meer, A. Spedalieri, J.P. Le Rouzic and J. Zoric, "Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform", Proc. ASWN '06, 2006.
[9] M. Strutterer, O. Coutand, O. Droeghorn and K. David, "Managing and Delivering Context-Dependent User Preferences in Ubiquitous Computing Environments", Proc. SAINTW '07, 2007.
[10] M. H. Williams, N. K. Taylor, I. Roussaki, P. Robertson, B. Farshchian, and K. Doolin, "Developing a Pervasive System for a Mobile Environment," Proc eChallenges 2006 – Exploiting the Knowledge Economy, IOS Press, 2006, pp. 1695 – 1702.
[11] S. McBurney, E. Papadopoulou, N. Taylor and M. H. Williams, "Adapting Pervasive Environments through Machine Learning and Dynamic Personalization", Proc. Int. Conf. on Intelligent Pervasive Computing (IPC-08), Sydney, Dec. 2008, in press.
[12] Dey, A.K., "Understanding and Using Context". Personal and Ubiquitous Computing Journal, 2, pp 139-172, 2001.
[13] M.H. Williams, I. Roussaki, M. Strimpakou, Y. Yang, L. MacKinnon, R. Dewar, N. Milyaev, C. Pils and M. Anagnostou, "Context Awareness and Personalisation in the Daidalos Pervasive Environment", *Proc. Int. Conference on Pervasive Systems (ICPS 05)*, Santorini, July 2005, pp. 98 – 107.
.