USER PREFERENCE MANAGEMENT IN A PERVASIVE SYSTEM SHOULD BE A TRUSTED FUNCTION

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

ABSTRACT
In developing a ubiquitous or pervasive system, the privacy of the user needs to be protected as much as possible. At the same time the system needs to take account of the user's wishes in taking decisions on behalf of the user by maintaining a set of user preferences for each user. This involves not only the management of such preferences but also monitoring the user and automatic learning of new preferences or updates to existing ones. The need for privacy may lead to a crucial decision in the design of a pervasive system as to whether or not to treat the components dealing with personalization and user preference management as trusted components. This paper discusses four of the problems that arise if these components are not treated as trusted. Two of these are dealt with relatively easily although the remaining two are more challenging. Overall, the paper demonstrates why it is important to have an integrated system in which this functionality is trusted.

KEY WORDS
Pervasive, User preferences, Privacy, Trusted.

1. Introduction

Since the notion of ubiquitous computing was first mooted in the early 1990s, both ubiquitous [1] and pervasive [2] computing environments have attracted growing attention. Pervasive computing addresses the problem of the increasing complexity associated with the rapidly increasing number of devices and services available to the user, and aims to provide the necessary support to enable the user to manage the situation. Although solutions have been developed for different aspects of the problem, many of the major problems facing ubiquitous and pervasive computing have yet to be solved satisfactorily.

In recent years a number of prototypes have been developed, focusing on different aspects of the problem. Although these types of systems are still very much prototypes or at the experimental stage of their development, some researchers believe that by 2020 these will be solved and this technology will be in the market place. In the meanwhile some of the global challenges of the next decade [3] lie in this area.

From an early stage the prototypes that were developed recognised the importance of personalization. As pervasive systems become more and more complex, they need to be managed in such a way that the user is in control. To do this it is essential to take account of user needs and preferences in decision making within such a system. This is necessary in order to produce a system that will be acceptable to the user. Equally important is the notion of privacy. Designing a system that ensures the latter could place constraints on the design of the other components in the system, such as the former.

Daidalos is a large research project funded by the European Union. Its target is the integration of a range of heterogeneous networks and devices and the creation of a pervasive system (service platform) [4] on top of this, which protects the user from the complexity of the underlying infrastructure while providing personalised and context aware services with minimal user intervention.

Two of the areas that have been studied in detail in Daidalos are personalization and privacy. In the process some attention has been given to the potential conflict between the two. One way in which this has been addressed is by designing and implementing two different prototypes based on different assumptions. In one the personalization and user preference management functions were treated as trusted functions and were integrated into the system. In the other these functions were not trusted and were isolated from the core of the pervasive system.

This paper discusses how personalization and user preference management are handled and some of the problems arising from treating them as untrusted functions that are isolated from the core. The results of this are being used to shape the architecture of another pervasive system which is being developed within the Persist project. The next section introduces briefly some aspects of personalization and pervasive systems. Section 3 briefly outlines the use of personalization and user

preferences in Daidalos. Section 4 describes problems arising from the implementation of the pervasive system in Daidalos when personalization and user preference management are not trusted. Section 5 summarizes and concludes.

2. Personalisation and Pervasive Systems

Personalisation is concerned with the adaptation of the appearance and/or behaviour of a system to meet the needs and preferences of each individual user. Recent research tended to focus on the Web, and was concerned with adapting the search or the presentation of information to suit individual users. There was also considerable interest in techniques that could be used in web services for marketing purposes. However, personalisation in pervasive systems is much wider than this.

The value of learning techniques to enhance personalisation was soon recognized and, as a result, a number of recommender systems were developed that would benefit the user during visits to e-commerce websites by recommending products that matched the user's preferences and budget.

To take account of the needs and priorities of the individual user, these need to be captured in a set of "User Preferences". In a pervasive system these will, in general, be context-dependent. These user preferences may take the form of a set of rules or possibly a neural network or Bayesian network. This paper is concerned with the case where the user preferences are a set of rules. The main challenge is to capture the user preferences in the required format and to refine and adapt these with time as the knowledge about the user is accumulated so that the resulting user profile adequately reflects the user's needs and priorities. By so doing one can automate some of the decision making in the system to relieve the user of this burden.

A number of pervasive systems prototypes include user preferences to assist decision making. Projects such as the Intelligent Home [5] and Blue Space [6] both rely on user-supplied preference information while others such as the Adaptive House [7], GAIA [8] and MavHome [9] monitor user actions and use learning techniques to discover and improve preferences. Yet other systems that use learning include Mobilife, Ubisec, etc.

3. The Daidalos Pervasive Platform

The basic functionality contained in the Daidalos pervasive system includes the following:
(1) Service Discovery and Selection.
(2) Service Composition.
(3) Session Management.
(4) Personalisation.
(5) Context Management.
(6) Security and Privacy.

The first prototype that was developed treated these six functions equally and provided a basic set of the functionality required to support the user in a pervasive environment. The architecture consisted of six modules although these were slightly different from the six basic functions listed above, and is described in [4].

In the second prototype different assumptions were made, including the fact that some of these six basic functions might be provided by different service providers. In this regard context management, personalisation and even service discovery and selection were identified as possible candidates that might be provided by software developed by different providers. As a result these modules were no longer treated as trusted components and the user's identity was concealed from them.

Within Daidalos user preferences are handled using a combination of three different strategies. Firstly, in order to assist the user with creating an initial set up of user preferences, a set of stereotypes is provided [10]. A stereotype consists of a typical set of user preferences that corresponds to a user with a typical pattern of behaviour. In particular, this may be useful in the case of individual services where cluster analysis can identify clusters in the options selected. When a stereotype is selected for a service, the relevant set of user preferences are loaded into the user profile, thereby enabling an initial profile to be created rapidly.

In addition to the use of stereotypes, user preferences can be created or modified by the user manually using a GUI, designed for the purpose. Although this is a useful way for the user to see what is stored and to tweak it to meet their requirements, it is not likely to be the main way of setting up and maintaining user preferences.

The third strategy used is that of automatic learning of preferences [11]. By monitoring the user's reactions to system decisions and recording the acceptance or rejection of these together with the current context of the user, a historical record is established in a database. To this learning mechanisms are applied to establish new patterns in behaviour and to create new preference rules or identify changes to existing ones.

The net effect of this is that the user is able to create an initial profile very rapidly and the system will then adapt the profile gradually with time as observation of the user's reactions to decisions lead to refinement of his/her preferences.

In order to protect the privacy of the user in Daidalos the pseudonym approach has been adopted in the form of a system of virtual identities (or VIDs) [12]. A virtual identity can be regarded as a form of user name although

it has additional properties relating to the confidential user data that it is prepared to release. Each user may have any number of virtual identities, which the user may use for different purposes.

4. Consequences of Treating Preferences as Untrusted

4.1 Effect on User Preferences

The first major effect of isolating the user preference management functionality from the core of the pervasive system concerns the linkage to VIDs. Whenever the user selects a stereotype or amends his/her preferences this is done through a VID. Likewise whenever the system observes a user action which might affect his/her preferences, this is linked to a VID. Thus from the point of view of the functionality for user preference management, it maintains a separate set of user preferences for each user VID.

However, this could be exceedingly frustrating for the user if he/she maintains a number of separate VIDs with overlapping preferences. For example, suppose that a user has five VIDs for different purposes but all have the same preferences regarding the selection and set up of voice call services. Not only must he/she select the appropriate stereotype and/or amend his/her preferences five times, but also the learning process will go through five times as many steps before it has learnt the correct preferences for all five VIDs.

To overcome this problem a system of indirection is used whereby the set of preferences associated with a VID is accessed via a pointer. The user can then specify to the Security and Privacy subsystem that any particular set of virtual identities should share the same set of preferences. This subsystem will then create an appropriate set of indirections so that the user profiles for each of these virtual identities point to the same set of user preferences.

Now suppose that VID1 and VID2 share the same set of preferences. If the user or a service operating on behalf of the user needs to access the preferences, the appropriate VID is passed to the User Preference Management function. The latter fetches the relevant set of preferences via the pointer associated with the VID, and performs whatever operation is required. In doing so, it knows nothing about the owner of the VID or whether or not the preferences associated with this VID are shared by any other VIDs.

On the other hand if the user amends any of the preferences associated with VID1, these will obviously also be amended for VID2. If the learning function detects user actions that might affect these preferences in either VID1 or VID2, these are recorded and when sufficient occurrences have been observed to establish a pattern, the preference is updated, independently of which VID was used for each individual action.

4.2 Evaluating Preferences

Once again to protect the privacy of the user, the user preferences themselves need to be protected. Since the preferences are context-dependent they may well express different preferences if the user is at home from when the user is at work. They may even express different preferences for locations that are particularly sensitive. They may contain preferences for particular service suppliers in certain contexts which the user would not wish other service suppliers to be aware of. And so on.

In order to avoid any sensitive information from being released to inappropriate users or services, the preferences themselves are not revealed to anyone except the user who owns them and who may inspect and change them via a GUI designed for this purpose. Whenever a service requires user preferences, these are evaluated and the resulting action, referred to as the preference outcome, is passed to the requesting service.

Thus as far as individual services are concerned, whenever they request user preferences, the current preference values are returned. What the alternatives may be and under what circumstances these might apply are not revealed.

However, while using a service for which a particular preference applies, the context of the user might change, and as a result the preference value might change. A simple example is the choice of network service to use. If the user is moving around or the traffic on the network changes, the QoS may drop, different networks may become available or the user may interject and the preferred choice of the user might change.

In order to handle this without revealing the user preferences, the user preference functionality provides this facility. When preferences are used in setting up or running a service, if they are context-dependent, then this aspect of the user's context is monitored as long as the service runs. If the context changes, resulting in a change in preference outcome, the service is informed of the new preference outcome, and it is left up to the service to decide whether or not to act on this.

4.3 Effect on Learning

The previous two sections identified solutions to the problem of maintaining the privacy of the user through the use of virtual identities and the concealment of the user preferences from all but the user him/herself. However, this section and the next consider other problems arising from the exclusion of the user preference

management (including learning) facilities from the core, which are less easy to solve.

The learning functionality is aimed at building up and refining the preferences of the user. To do this it must monitor the decisions taken by the user and the context in which the decisions are taken in relation to specific system tasks.

A simple example and one which is most important for pervasive systems, is that of selection of services. When the user requests a service, a service discovery process is invoked to find whatever services may be available that could be used to meet the user's request. This is followed by or integrated with a service selection process, which applies the user preferences to the discovered services. These can be used to *filter* the list by removing services that the user definitely does not want, and to *rank* the list by ordering the services according to the preferences of the user.

The result is a ranked list of services which could satisfy the user's request. It may be that a single service is sufficient to satisfy the request or it may be that two or more services need to be composed together to meet the request. Whatever the case, once a selection has been made, it is necessary to check whether these services are in fact still live and available. The fact that the service discovery process has discovered them at some point in the past does not guarantee that they are still live and available for use – for example, some other user may have started using the service since then.

Thus at this point the system needs to test the services selected for the composed service, and if not all are available, it must systematically try all possibilities from the list in order until it eventually finds a composition that is live and available, in which case the composition is executed.

The user is informed of the final choice of service made and is given the opportunity to stop the composition if the choice is not what the user wanted. If he/she does so, a different selection will be made.

There are three problems with this approach. Firstly, the service has already begun to execute when the user is given the opportunity to stop it. This means that if it were the wrong choice, the user would have to pay for starting to execute it. Secondly, it may intrude on the privacy of the user if the wrong service were started as it may give information to the particular service provider about the user.

The third problem is that the learning function does not have a complete picture of what has happened in the selection process (e.g. why other preference options have not been presented to the user) nor does it know which part of a composition was unacceptable to the user.

4.4 User Preferences for Managing Privacy

Having pointed out earlier that user preferences and privacy are naturally in conflict with each other, there is one area where a close link between the two is required. As already mentioned, when the user makes a request to the system, the system discovers the most appropriate services needed to meet that request and composes them into an executable service to handle the request, and then executes the composed service. In doing this more than one VID may be required. First one needs a VID associated with the user request to perform service discovery, selection and composition. Then one needs at least one VID for the composed service, although more than one VID may be used if required for the component services.

Now once again decisions are required. Again one can make the simple assumption that the user will always select the appropriate virtual identity before requesting any service. However, this can become an arduous task for the user, especially if the number of virtual identities grows. One might argue that this may be a good reason to restrict the number of virtual identities to a very small number but this defeats the purpose of having a flexible system of virtual identities.

The situation is more complex if one takes account of changing context conditions. Consider the case of a mobile user who is using a network service. As the user moves around he/she may need to change to a different network service (because of falling Quality of Service on the current network due to wireless reception difficulties, increased network traffic, etc., or simply the availability of a more preferable network service). It is not sensible to interrupt the user to ask whether he/she wants to change and, if so, what virtual identity should be used.

Thus in order to provide a user-friendly pervasive environment the system itself should manage the automatic selection of virtual identities wherever possible, only resorting to user decision or intervention when absolutely necessary. In order to do this, whenever a VID is required for a service, the VID management functionality executes the following approach:
(1) It determines what private user data are required by the service and negotiates via a set of privacy policies what it is prepared to release to the service.
(2) From this a set of potential VIDs that can provide this data is identified.
(3) The most suitable VID is selected from this list.

In order to do this one needs to know what VID the user would prefer to use in different circumstances. In other words, one needs to know the user's preferences. However, this set of user preferences needs to be treated differently from the rest of the user profile. Although their format is essentially the same, the action performed (the

selection of a virtual identity) is highly confidential. One would also like the system to learn the user's preferences as it does for other preferences of the user.

Now one has an even bigger problem with the assumption that user preference management should not be trusted. In order to provide the functionality described, two alternatives are possible, viz.
(1) One could create a copy of the user preference management functionality (including the learning functionality) and adapt it for privacy management. This can then be incorporated into the core and contained completely within the Security and Privacy subsystem and used solely for this purpose. In other words it becomes a trusted component. This would ensure privacy although at the expense of a considerable amount of duplicated functionality.
(2) One retains the user preference management functionality outside the core, but conceals the actual VIDs from it. One possibility may be to use cryptographic techniques. These might be applied to the actions pertaining to the selection of virtual identities before such information is passed to the preference management subsystem, and likewise the information returned may be decrypted, thereby ensuring the privacy of the user. Thus when the privacy functionality wants to create or update a user preference, it passes the information together with encrypted VID information to the user preference management functionality to process and store. Likewise, when it needs a user preference, it invokes the user preference management functionality, and decrypts any VID information returned. The preference management and learning subsystem can treat the resulting preferences in the same way as for any other service without understanding the actions.

Both approaches lack parsimony and cohesion. In addition to this the second approach is unduly cumbersome whilst leaving unresolved problems for the learning functionality. The first approach is preferable but it breaks the assumption that personalization and user preference management should not be trusted.

5. Conclusion

This paper is concerned with one of the major issues relating to the design of pervasive or ubiquitous systems, namely the conflict between personalisation and privacy. On the one hand one has the problem of building up a relevant set of user preferences to assist decision making in the system. On the other hand one has security mechanisms to protect the privacy of the user. These two can be in direct conflict. One possible consequence is that the personalisation and user preference management functionality (including the learning of user preferences) is treated as untrusted and isolated from the core. The effects of such a decision have been studied in the Daidalos pervasive system.

The first problem that arises is that the user preference functionality (including learning) must be prevented from knowing anything about the real identity of the user and may only operate through VIDs. The second concerns how the system can protect the user preferences so that only preference outcomes are seen by any other user or service. Both of these can be readily addressed.

On the other hand the untrusted approach causes major problems for the automatic learning of user preferences. Not only is this functionality hampered by this assumption, but the user may also be inconvenienced and may find him/herself paying unnecessarily for services.

In addition one has a major problem in the automatic selection of virtual identities for use in the privacy subsystem. Unless the user selects these whenever they are needed, one needs to rely on the user preference management functionality. Either one resorts to some form of cryptographic technique to enable the continued use of the user preference management functionality outside the core for this purpose (with consequential restrictions) or one relaxes the restriction and duplicates functionality within the core to carry out this function.

Thus the idea of treating personalisation and user preference management functionality as untrusted should be avoided if at all possible, as the disadvantages encountered outweigh any advantages of such an approach.

Finally, of all a user's personal preferences, those pertaining to his/her privacy requirements are perhaps the most important to him/her. It therefore seems eminently sensible to ensure that preference management systems are designed around this type of user preference, rather than handling them in a non-homogeneous and ad hoc fashion at a later stage.

The result of this study is being used in tailoring a new pervasive architecture in the Persist research project (*PERsonal Self-Improving SmarT spaces*).

Acknowledgements

The research leading to these results has received funding from the European Community through the research project Daidalos (Sixth Framework Programme) and are contributing to decisions in the project Persist (no. 215098) in the Seventh Framework Programme. The authors also wish to thank all colleagues in the Daidalos project developing the pervasive system. However, it should be noted that this paper expresses the authors' personal views, which are not necessarily those of the Daidalos consortium. Apart from funding these two projects, the European Commission has no responsibility for the content of this paper.

References

[1] T. Kindberg, & A. Fox, System Software for Ubiquitous Computing, *IEEE Pervasive Computing*, *1*(1), 2002, 70-81.

[2] M. Satyanarayanan, Pervasive computing: vision and challenges, *IEEE PCM*, *8*(4), 2001, 10-17.

[3] The UK Grand Challenges Exercise. Available: http://www.ukcrc.org/grand_challenges/

[4] M. H. Williams, N. K. Taylor, I. Roussaki, P. Robertson, B. Farshchian, & K. Doolin, Developing a Pervasive System for a Mobile Environment, *Proc. eChallenges 2006 – Exploiting the Knowledge Economy*, IOS Press, 2006, 1695 – 1702.

[5] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan & S. Zhang, XQ.: The Intelligent Home Testbed, *Proc. Anatomy Control Software Workshop (Autonomous Agent Workshop)*, 1999, 291-298.

[6] S. Yoshihama, P. Chou & D. Wong, Managing Behaviour of Intelligent Environments, *Proc. First IEEE Int. Conf. on Pervasive Computing and Communications (PerCom '03)*, 2003, 330-337.

[7] M. C. Mozer, Lessons from an Adaptive House, in D. Cook & R. Das (Eds.), *Smart Environments: Technologies, protocols and applications*, 2004, 273-294.

[8] B. D. Ziebart, D. Roth, R. H. Campbell, & A. K. Dey, Learning Automation Policies for Pervasive Computing Environments, *Proc. 2nd Int. Conf. on Autonomic Computing (ICAC '05)*, 2005, pp. 193-203.

[9] M. G. Youngblood, L. B. Holder & D. J. Cook, Managing Adaptive Versatile Environments, *Proc. 3rd IEEE Int. Conf. on Pervasive Computing and Communications (PerCom '05),* 2005, 351-360.

[10] E. Papadopoulou, S. McBurney, N. Taylor, M.H. Williams, & G. Lo Bello, Adapting Stereotypes to Handle Dynamic User Profiles in a Pervasive System, *Proc. 4th Int. Conf. on Advances in Computer Science and Technology (ACST 2008)*, Langkawi, Malaysia, 2008, pp. 7-12.

[11] S. McBurney, E. Papadopoulou, N. Taylor, & M. H. Williams, Adapting Pervasive Environments through Machine Learning and Dynamic Personalization, *Proc. Int. Symp. On Parallel and Distrib Processing With Applications/Conf. on Intelligent Pervasive Computing (ISPA-08/IPC-08)*, Sydney, 2008, pp. 395-402.

[12] J. Girao, A. Sarma, & R. Aguiar, Virtual identities - a cross layer approach to identity and identity management, *Proc. 17th Wireless World Research Forum*, Heidelberg, Germany, November 2006.