A Dynamic Approach to Dealing with User Preferences in a Pervasive System

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

Abstract

User preferences have an important role to play in decision making in pervasive systems, and most systems incorporate some form of preferences, albeit that different forms and approaches are used in different implementations. It is generally accepted that user preferences need to be context-dependent, although the process of applying user preferences to personalize decision taking in the system tends to be treated as a static process that is carried out at the outset when a service is started and not referred to again unless the user specifically instructs a change. However, there are situations where a more dynamic approach to dealing with user preferences is called for. This has been investigated as part of the Daidalos pervasive system, and this paper reports on the approach used and implemented in this system.

1. Introduction

Following Weiser's predictions [1] that the environment around the user will soon be filled with microscopic devices, mobile or stationary, that will aid the user in his/her everyday life, developments in sensor technologies, communication technologies, mote [2] and speck [3] technologies, etc., are enabling these predictions to be realized, although not yet at the scale that Weiser had in mind. The growth in the number of sensors and devices in the user's environment coupled with the increasing number of different kinds of communication networks will give rise to an even larger array of services for the user to choose from. The problem for the user of this increasing complexity is that it will become overwhelming and unmanageable. This is the problem that pervasive computing [4] seeks to address by developing an intelligent environment to support the user and to enable him/her to control and manage this situation [5]. The importance of this problem is reflected in the fact that some of the “global challenges” of the next decade [6] have been identified in this area.

In order for such a system to be acceptable to the user and to hide the complexity of the underlying infrastructure from the user, it is essential that the system should take decisions on behalf of the user. This can only be done if the system knows what the user would prefer, i.e. it maintains a set of user preferences for each user and uses these to personalize the decision making processes within the pervasive system. Different approaches are used by different systems to achieve this.

The simplest approach to handling user preferences in a pervasive system is a static one in that once the user preferences have been applied to a service they are not referred to again unless the user specifically instructs a change. However, there are situations where a more dynamic form of personalization is called for. For example, suppose that a user has a voice call which involves using a network service and connects to some device (e.g. mobile phone). If the user moves around, the Quality of Service (QoS) of the network may drop and it may be desirable to change to a different network service if one is available. If the user gets into his/her car, he/she may prefer the call to be redirected to a phone installed in the car as part of the car’s multimedia system. And so on.

The importance of handling preferences dynamically, especially for mobile users, is recognised in the Daidalos system. Daidalos is a large European research project whose aims include the development of a pervasive environment [7], based on a pervasive service platform, which will provide the appropriate infrastructure to support a wide range of personalized context aware services in a way that is easy for the end-user to manage and use.

Consequently, a general approach has been adopted in Daidalos, which caters for dynamic as well as static handling of user preferences. In this case the effects of personalization on a particular service may change during a session if the context of the user changes. This paper reports on the dynamic approach to dealing with
user preferences, and how this is realised within the Daidalos system.

The next section provides a brief overview of personalization and user preferences in pervasive environments and how this affects some current systems. Section 3 briefly describes the structure of the Daidalos pervasive system while Section 4 focuses on personalization in Daidalos. Section 5 provides some detail on the concept of dynamic preferences and how they are implemented in Daidalos. Section 6 provides a brief summary and conclusion.

2. Personalization and Pervasive Environments

The need for individual user preferences and their application in decision making in a pervasive environment is now widely accepted. The process of creating, maintaining and applying user preferences in decision making is sometimes referred to as personalization, since it has the effect of tailoring the system’s behaviour to the individual needs and wishes of the user so that it appears or acts differently for different users or for the same user under different circumstances. Although most research on personalization has tended to focus on the problems of retrieval of information on the Web (e.g. by adapting user queries to improve the relevance of the answers returned) or on controlling the layout and presentation of output from applications [8], within pervasive systems there is scope for a much wider interpretation of the term. Some examples are given in section 4.

Thus far the work done on different pervasive systems has incorporated personalization techniques with varying degrees of success. Early developments such as Microsoft’s Easy Living project [9] and MIT’s Intelligent Room project [10] concentrated on the use of context information rather than on user preferences – producing a context aware rather than a personalized approach. However, the importance of incorporating user preferences into the decision making was soon identified and other projects such as the Intelligent Home [11], Blue Space [12] and AURA [13] implemented both context awareness and personalization - although they relied on user input of preference information, resulting in minimal sets of user preferences.

The problem of capturing and maintaining user preferences was soon recognized and the need to assist the user in this process was established as an important requirement for future systems. As a result, projects such as the Adaptive House [14], GAIA [15] and MavHome [16] place automation as their key goal with the intention of completely mitigating explicit provision of user information for personalization purposes. Monitoring and learning algorithms unobtrusively gather preferences and environment information such as user movement and actions which are used to predict future movements and actions. Based on predictions, environments are automatically adapted by applying user preferences. However, removing the possibility for user input removes all user control which may lead to confusing or frustrating situations, perhaps discouraging system use.

The Synapse project [17] takes steps to find a balance between automation and user control by operating in active or passive mode. If a preference has a probability above some threshold the preference is applied automatically in active mode, whereas passive mode consults the user with suggestions before preference application. However, preferences may take time to switch between active and passive modes and although there is the possibility of more accurate personalization one could speculate about the helpfulness of this solution as the user could possibly be inundated by pop-up messages.

More recently, projects such as Ubisec [18], Spice [19] and Mobilife [20] aim to provide the user with personalized services in a global environment. Once again preferences are applied automatically but disparately these projects have attempted to provide more accurate personalization by implementing more responsive implicit personalization. Mechanisms are implemented which respond rapidly to changes in the user’s behaviour patterns and update the set of user preferences in real time. A potential disadvantage of such mechanisms is that they do not distinguish between temporary changes in user behaviour arising from particular transitory circumstances and established user preferences derived from long-term behaviour patterns.

3. The Daidalos Pervasive Platform
Daidalos is a large research project, involving more than 40 partners and funded by the European Union. Its target is the integration of a range of heterogeneous networks and devices and the creation of a pervasive environment (based on a pervasive service platform) on top of this. This protects the user from the complexity of the underlying infrastructure while providing personalized and context aware services with minimal user intervention. It has a specific focus on mobile users. The research conducted is now in its fifth year, and was divided into two phases with slightly different objectives.

The basic functionality contained in the Daidalos pervasive service platform includes the following:

1. Service Discovery and Selection. Whenever the user requests a service, the system relies on this functionality to discover appropriate services that might meet the user’s request fully or in part, and to filter and rank them according to the user’s preferences.

2. Service Composition. This is responsible for assembling the selected services to create a composed service that meets the user’s request.

3. Session Management. This has the task of managing the user’s session created by the request.

4. Personalization. This is responsible for handling personalization at various points in the process of providing user services. These include the filtering and ranking of services in the process of discovery and selection, personalizing third party services, learning new and managing existing user preferences, etc.

5. Context Management. This manages a range of different types of context data both pertaining to the user (e.g. location) and to the available services and resources.

6. Security and Privacy. This is responsible for ensuring user privacy. The approach used is based on pseudonymity [21, 22] using virtual identities, which allows the user to control knowledge on who is running what services and what access is permitted to user data.

In the first phase of the project the development focused on providing a basic set of functionality to support the user in such an environment, although the work done to ensure the privacy of the user was limited. In this phase the six basic functions were treated equally. The architecture consisted of six modules although these were slightly different from the six basic functions as shown in Fig. 1 and described in [7].

By contrast, in the second phase there is a much greater focus on privacy. It is also assumed that some of the functions might be provided by different service providers. This has a serious consequence if modules are no longer trusted components and the user’s identity needs to be concealed from them.

Likewise in the second phase there is greater emphasis on the idea of dynamic handling of user preferences through the personalization component of the Daidalos platform. The next section provides a brief overview of personalization and user preferences in Daidalos.

4. Personalization and User Preferences in Daidalos

Within Daidalos different strategies are being evaluated for the storage and manipulation of user preferences. One of these is a rule-based approach. This approach has the advantage that the user can inspect and modify his/her preferences at any time.
The user preference rules that are established using the rule-based approach are used to personalize various different decision processes within the pervasive system. These include:

1. Selection of services in response to a user request. When a user requests a service there may be a number of different alternatives that are possible candidates that might be used to satisfy this request. For example, if he/she wants to print a document there may be a number of different print services to choose from, varying in cost to the user, quality, distance from the user, etc. If he/she wants a news service, options may include broadcast TV, Web news services, mobile phone services, etc. Instead of asking the user to choose each time, his/her preferences can be used to make the decision – which may depend on a variety of factors such as the user’s location, time of day or current activity.

2. Personalization of individual services through parameters. Both third party services and services internal to Daidalos (referred to as Daidalos Enabling Services) can be adjusted to meet the user’s needs by passing appropriate values for parameters. For example, if the user is receiving a telephone call, the volume could be adjusted automatically according to user preferences depending on where the user is located (at home, at work, in the car, in the street).

3. Personalized call/message delivery and redirection. With the potential to access different devices and networks, it is possible to connect incoming (VoIP) telephone calls or messages to different devices depending on the user’s preferences. This will depend on where the user is located, the time of day and possibly on the identity of the caller or other factors. For example, when the user is at home, he/she can have business calls (identified through the caller id) redirected to his/her home telephone during working hours and to a voicemail service outside of working hours. At the same time personal calls can be redirected to another member of the family if the caller id is recognized as being relevant to them (e.g. daughter’s boyfriend). If the user is traveling in his/her car, an incoming telephone call may be directed to the car phone if the caller is deemed to be important enough. And so on. This was studied in detail in the first phase of Daidalos [23].

Personalized call delivery also entails converting the format in which the communication arrives. For example, a user can be located in a place where loud noises prevent him/her from being able to talk or listen on the phone and therefore, a sound to text service can be used to deliver the communication.

4. Selection of virtual identities to protect the privacy of the user. This is an area that has received particular attention in the second phase of Daidalos.

The initial design of the virtual identity approach used in Daidalos is reported in [24]. In this approach the user may have a number of different virtual identities that he/she can use to protect his/her real identity and thereby ensure privacy. Assistance in the selection of virtual identities is based on a set of user privacy preferences which can be acquired using a combination of user querying and automatic learning.

A common format has been adopted for the user preferences for all of the above situations. In each case, the requesting service, Daidalos Enabled Service (DES) or third party service, is treated in the same way. Thus, Service Filtering and Service Ranking preferences look the same as the preferences for a third party video streaming service.

However, each of these different areas of personalization has its own set of preference rules for each user – i.e. separate preference rules for service selection, for setting personalizable parameters, and so on. Furthermore, there is a separate preference rule for the selection of each type of service or the setting of each personalizable parameter for a service. The format adopted for such rules consists of a simple “if-then(-else)” rule. The condition part of this rule is a Boolean expression which may involve checks on context attributes such as location, possibly combined with AND, OR and NOT operators. Each then-part or else-part may be either an action or a nested if-then(-else) statement.

An example of parameter setting might be:

IF location = “home” AND status = “free”
THEN SET volume TO 10
ELSE SET volume TO 5

which sets the volume of the device according to the location and status of the user. Similarly, the following preference rule

IF location = “home”
THEN printer.distance < 10
ELSE IF location = “work”
THEN printer.distance <50
FI

sets the constraint on printer.distance according to where the user is.

Actions are represented in a preference in the same way as personalizable parameters by setting their parameter names as “action”, compared to normal personalizable parameters which would have parameter names such as “volume”, “colour” and other similar attribute names. Note that the Personalization subsystem does not itself call the methods of the
service interfaces. Instead, it is responsible for evaluating the preference rules for a user and delivers to a service the outcome of evaluating the user preference in the form of a Java object. This is referred to as the preference outcome. The service can then decide, based on the outcome and its own knowledge about the state and other internal factors, whether or not to act on the outcome.

As mentioned in section 2, the importance of establishing user preferences is a priority in Daidalos as with other pervasive systems. For this reason automatic learning of preferences plays an important role and this is discussed in a separate paper.

5. Dynamic Personalization

In most applications personalization is treated as a static process in which user preferences are applied to the decision making process, and once the decision is taken, are not referred to again unless the user intervenes. However, when combining personalization with context awareness, changes in context may affect the user preferences and hence the outcome of the decision making process.

Consider the situation where a user makes a request and the system creates a composed service to meet this request. In the case of some services, once the composed service starts to execute, no further action is required from the point of view of the Personalization component. In other cases this is not so. As mentioned previously, if a mobile user is engaged in a voice call directed to his/her mobile phone then as he/she moves around, changes in his/her context could mean that decisions taken when the service was started are no longer optimal with regard to the user’s preferences.

For example, if the user gets into his/her car, he/she may prefer the call to be directed to a phone installed in the car as part of the car’s multimedia system. If the Quality of Service (QoS) of the network falls, it may be desirable to change to a different network service if one is available. And so on.

This section describes how this is handled in the context of the Daidalos system.

5.1 Personalization Subsystem in Daidalos

The idea adopted in Daidalos is that personalization of pervasive services goes beyond any traditional static personalization. The main requirements for dynamically personalizing pervasive services are that user preferences are constantly evaluated and updated based on the user’s context, behaviour and other environmental factors and services are recomposed and re-personalized based on the newly inferred preference outcomes whenever a relevant change in the conditions arises.

In order to handle this, the Personalization Subsystem in Daidalos includes the following three components (see Fig. 2):

1. Preference Manager. This is responsible for managing the user preferences. When a new preference is created or an existing one updated (either manually by the user or automatically by the Learning Manager) the Preference Manager is responsible for updating the preference set appropriately. When a module requests a preference, the Preference Manager evaluates the preference using the current context and returns as result the preference outcome.

2. Preference Condition Monitor. When a service is started on behalf of the user, if the service is affected by any context-dependent preferences, this component will receive any context changes and check whether

Figure 2. Part of the architecture of the Pervasive Service Platform used in the second phase of Daidalos
they affect any preference outcome relating to this service while it is running.

(3) Action Handler. This is responsible for passive monitoring, including the detection of events signaling a change in context of the user.

In addition to these there are several other components, including a Learning Manager, but these will not be described here as they are not relevant to this paper.

5.2 User Request for Service

The first step in personalization is a conventional one and the actions here are the same as the static case. When the user requests a service, this request is processed by the Pervasive Service Manager. It invokes Service Discovery to discover any potential services that might be available that can be used to satisfy the user request. The list of potential services produced by Service Discovery is then filtered to remove any that definitely do not satisfy the user’s preferences. The resulting list is ranked according to the user’s preferences, taking into account his/her current context.

The resulting ranked list of services is used by the Service Composition component to assemble a composed service that will meet the user request. In the simplest case a single service might suffice and this becomes the composed service. In more complex cases two or more services need to be linked together to create the composed service.

The composed service is passed to the Session Manager to run. When the composed service is ready to execute, the user is informed of the services that have been selected on his/her behalf and is given the opportunity to intervene if these are not acceptable. If the user does not intervene the composed service proceeds to execute. In either case the Learning Manager is notified so that it can maintain a record of what the user found acceptable or not.

5.3 Preparing for Execution

Once the composition is complete and all the services are running, the Pervasive Service Manager notifies the Action Handler of the new composition. This information eventually reaches the Preference Condition Monitor (PCM). The latter maintains a list of running services and for each service a list of conditions that affect the preferences of this service. For each of these conditions, the PCM is registered with the Context Manager to receive constant updates to these.

When the information about the new composition reaches the PCM, this list of conditions is updated and the PCM will request the preference conditions of the preferences of the services in this new composition from the Preference Manager (PM). The PM, which is responsible for user preference maintenance and evaluation, fetches the user preferences of the newly running services, adds them to its preference cache and returns the conditions enclosed in them to the PCM. In the case of a user preference of a service being context independent, the PM will return nothing and the PCM will have no further contact with the service. Upon receiving the conditions, the PCM will register with the Context Manager for any changes in these particular attributes.

At this point, each individual service will start to request preference outcomes from the PM. The PM, in turn, will evaluate the preferences of these against the current values of any context attributes and the outcome will be returned directly to the services.

5.4 Preference Condition Monitoring

Now whenever there is a change in the context of the user, which affects any of these attributes, the PCM is notified of this. The PCM will identify which preferences need to be evaluated by cross referencing its list and will request a reevaluation of these from the PM. The PM then evaluates the preference rules to check whether this results in any change in the preference outcome for any of the attributes. If the preference outcome is not affected, no further action is required.

If the preference outcome for one or more of these attributes does change, the PM will return these to the PCM which will perform one of three possible actions:

(1) If the change affects a personalizable parameter of one of the services in a composed service, the PCM sends a message to this service notifying it of the change. It is then up to the service to take whatever action it deems necessary.

(2) If the change affects the choice of individual services (apart from network services) that have been linked together to form the composed service, the PCM triggers a recomposition. This means that Service Discovery will be invoked, and a new list of potential services will be produced, which have been filtered and ranked according to the changed context and preference outcomes. Service Composition then composes these to produce a replacement for the original composed service.

(3) If the change affects a network service, this is handled slightly differently from other services. If it is necessary to change a network service (e.g. because of
falling QoS), this needs to be done very rapidly and cannot wait for a full recomposition. In this case the PCM communicates with the underlying layers via a component located in the transport layer which itself communicates it to the network layer to notify the appropriate module of any change in preference outcome that might affect the choice of network service. The network layer then decides whether or not to act on this based on information gathered from the network and the available device resources, and if it does act, it switches network with minimal interruption to the services using it.

It is also possible for the user to change his/her preferences during the course of a session by accessing the User Preference GUI. In this case, if a change affects a preference of a running service, the new preferences are cached and the old ones are discarded. The preferences are again re-evaluated to act upon the user’s changed requirements.

6. Conclusion

User preferences have a key role to play in pervasive systems. Their application to decision making processes is referred to as personalization as it tailors the behaviour of the system to the needs of the individual user. This is essential to relieve the user of the potentially overwhelming burden of selecting and, where necessary, changing services, devices and networks, as well as controlling individual services.

The simplest and most obvious way of handling user preferences in pervasive systems uses a static form of personalization in which once the user preferences have been applied to a service, they are not referred to again unless the user specifically instructs a change. This is all that is needed in many cases. However, in cases where the context of a user may be changing, especially those of mobile users, a more general approach is sometimes required. For example, if the user is engaged in a telephone call, changes in location might require a change in the network used or the device on which the call is delivered. Handling this requires a more dynamic form of personalization.

Since the Daidalos pervasive system is particularly aimed at the mobile user, the idea of dynamic personalization is an important requirement of the system. For this reason Daidalos caters for both static and dynamic personalization, with the result that the effects of personalization on a particular service may change during a session if the context of the user changes.

The initial prototype for the Daidalos pervasive system produced in the first phase of the project was demonstrated in 2005. The ideas described in this paper extend this functionality and implementation is almost complete and will be demonstrated by the end of 2008.

7. Acknowledgements

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. The authors 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. Note that apart from funding these two projects, the European Commission has no responsibility for the content of this paper.

8. References

Int. Conf. on Pervasive Computing and Communications (PerCom '03), 2003, pp. 330-337.


