Two Approaches to Handling Proactivity in Pervasive Systems

Elizabeth Papadopoulou[1], Yussuf Abu Shaaban[1], Sarah Gallacher[1], Nick Taylor[1] and M. Howard Williams [1]

[1] School of Maths and Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, UK
{E.Papadopoulou, ya37, S.Gallacher, N.K.Taylor, M.H.Williams}@hw.ac.uk

Abstract. A key objective of a pervasive system is to reduce the user's administrative overheads and assist the user by acting proactively on his/her behalf. The aim of this paper is to present some aspects of how proactivity is handled in the approaches used in two different pervasive systems. The Daidalos system provides proactive behaviour based on the assumption that the user is responsible for requesting services and that proactivity is restricted to selecting and personalising these based on the user's preferences. The Persist system uses an extension of this approach combined with an analysis of user intent. The idea behind the latter is that, if the system knows what the user will do next, it can act on the user's behalf, initiating the actions that the user would normally perform. User intent predictions and those produced by the user preferences are used to determine the final action to be taken.

Keywords: Pervasive computing, Proactivity, User Preferences, User Intent.

1 Introduction

Pervasive computing [1, 2] is concerned with the situation where the environment around a user is filled with devices, networks and applications, all seamlessly integrated. As developments in communications and in sensor technologies are accompanied by a large expansion in services available, the result will soon become unmanageable and it is this problem that pervasive computing seeks to address by developing an intelligent environment to control and manage this situation [2].

To hide the complexity of the underlying system from the user, the system needs to take many decisions on behalf of the user. This can only be done if it knows what the user would prefer, i.e. it maintains a knowledge base that captures user preferences for each user and uses these to personalize the decision making processes within the pervasive system. This may be further enhanced with mechanisms for determining user intent and predicting user behaviour. Without this it is difficult for a pervasive system to identify accurately what actions will help rather than hinder the user. This is one of the major assumptions underpinning most pervasive system developments.

In tackling the problem of developing ubiquitous and pervasive systems over the past decade, different research projects have adopted different assumptions and explored different approaches. As a result one class of system to emerge is that of the fixed smart space. This is generally focused on intelligent buildings – systems geared towards enhancing a fixed space to enable it to provide intelligent features that adapt to the needs of the user. On the other hand there are also systems that are focused on the mobile user, where the requirement is for access to devices and services in the user's environment wherever he/she may be. A number of prototype ubiquitous/pervasive systems have been emerging in recent years. Examples include [3 – 9]. Another example was the Daidalos project [10].

A novel approach that is currently being investigated in the research project, Persist, is that of the Personal Smart Space (PSS). The latter is defined by a set of services that are located within a dynamic space of connectable devices, and owned, controlled or administered by a single user or organisation. This concept has the advantage that it provides the benefits of both fixed smart spaces and mobile pervasive systems.

This paper is concerned with the problem of handling proactivity in a pervasive system and it describes the approach used in the Daidalos platform and compares it with that being developed for the Persist system. The former is based purely on user preferences whereas the latter uses a combination of user intent predictions and user preferences to provide input to decisions on proactivity. By incorporating user intent predictions, the system can identify situations that will require it to perform operations well in advance, thereby further reducing user involvement and enhancing user experience.

The remainder of this paper is structured as follows. The next section looks at related work on proactivity in pervasive systems. Section 3 introduces the Daidalos and Persist systems. Section 4 describes the approaches used in these two systems. Section 5 concludes and details future work.

2 Related Work

One of the major assumptions of pervasive systems is that they are adaptive to the needs of the individual user and able to personalise their behaviour to meet the needs of different users in different contexts. To do this such a system must retain knowledge about the user's preferences and behaviour patterns in an appropriate form and apply this in some way. This may be done proactively by identifying what actions the user might wish to take and performing these actions on the user's behalf.

Research on the development of fixed smart spaces identified the need for some form of proactivity from the outset. This was true both for smart homes (e.g. Intelligent Home [3], MavHome [4]) and smart office applications (e.g. Gaia[5], i-Room [6]). As interest extended to mobile users, nomadic applications such as Cooltown [7] used similar technologies.

Initial systems were based on the use of user preferences that were entered manually by the user. However, building up preferences manually is an arduous undertaking and experience has shown that the user soon loses interest and the resulting preference sets are incomplete and not very useful. In the case of the Aura project [8] manual input from the user was taken a step further when user intent was incorporated into the process of determining when to perform a proactive action and what action to carry out, as the user was required to provide not only user preferences but also information such as the user's current task.

As a result systems sought alternative approaches to creating and maintaining user preferences automatically. This inevitably involved some form of monitoring of the user's behaviour followed by some form of learning applied to the accumulated data. This approach was adopted in both fixed smart space developments (such as the MavHome [4] project) as well as mobile applications, e.g. Specter [9]. Another approach using Bayesian networks or Hidden Markov Models rather than rule based preferences to capture user behaviour and represent user needs was adopted by the Synapse project [10], which matches the context and current user task against past learnt patterns to select the most appropriate service for the user. Another system that uses context history for progressively learning patterns of user behaviour in an office environment is described by Byun et al [11],

3 Architectures of Daidalos and Persist Pervasive Systems

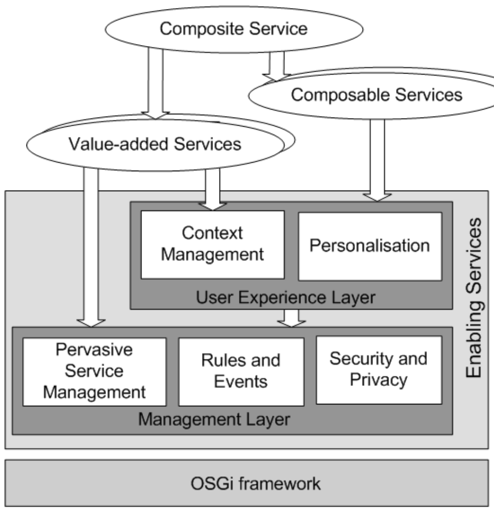

Fig. 1. Architecture of Daidalos Pervasive Services Platform.

The Daidalos project [12] developed a pervasive system based on the mobile user. The Pervasive System Platform (illustrated in Fig. 1) includes a personalisation and preference management subsystem (including learning) which implicitly gathers and manages a set of preferences for the user by monitoring user behaviour and extracting preferences from the monitored user behaviour history. This pervasive system was successfully demonstrated in December 2008. The main focus of the personalisation subsystem was to help in the selection and personalisation of services. By so doing the system was able to personalise the user's environment in an unobtrusive and beneficial way (based on previous user behaviour).

The Persist project is another European research project which started in April 2008. It aims to create a rather different form of pervasive system based on the notion of a Personal Smart Space (PSS) [13]. A PSS is defined by a set of services that are running or available within a dynamic space of connectable devices where the set of services and devices are owned, controlled, or administered by a single user or organisation. In particular, a PSS has a number of important characteristics.

- The set of devices and services that make up the PSS have a single owner, whether this is a person or an organisation.
- A PSS may be mobile or fixed.
- A PSS must be able to interact with other PSSs (fixed or mobile). To do this a PSS must support an ad-hoc network environment.
- It must be able to adapt to different situations depending on the context and user preferences.
- It must also be self-improving and able to learn from monitoring the user to identify trends, and infer conditions when user behaviour or preferences change.
- Finally it must be capable of anticipating future needs and acting proactively to support the user.

The architecture of a PSS in Persist is shown in Fig. 2.

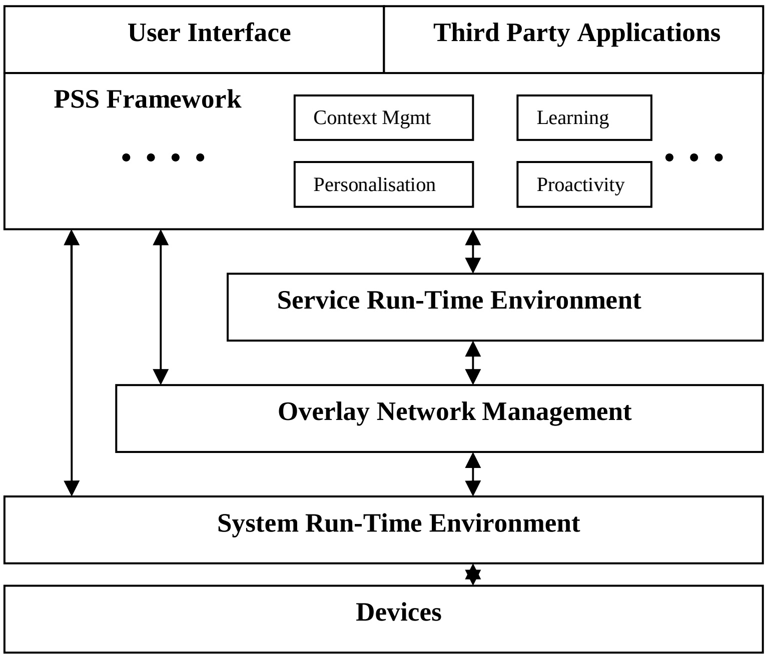

Fig. 2. The high level architecture of a Personal Smart Space

4 Approaches to Proactivity

This section describes two slightly different approaches used to handle proactivity in the Daidalos and Persist projects.

4.1 Proactivity in Daidalos

Proactivity in Daidalos is based on the assumption that the user will always initiate services and, where necessary, terminate them. Thus if a user wanted some service type, he/she would request this via a high-level GUI. However, because of the potentially wide range of services that may be selected to satisfy the user's request (which may, in turn, depend on a choice of devices and/or networks available), user preferences are used to help make this selection.

For example, if the user requests a news service, this request could be satisfied in a variety of ways. If the user is in his/her living room at home, the most obvious way to provide this might be by selecting an appropriate channel on the television set. If the user is at work, the best option might be to connect to a Web-based news service through the user's work PC. If the user is walking around town at the time of the request, the best option might be to connect to a phone-based service through his/her mobile phone. In all cases context-aware user preferences are used to select the most appropriate service to satisfy the high-level request.

Once a service has been selected, user preferences are used to customize it to meet the user's needs. For example, different levels of service may be available (Gold, silver, bronze) and depending on whether the user is using the service from work or from home, a different level may be chosen. The Quality of Service (QoS) acceptable in a communications service (e.g. VoIP) may depend on the user's context. Even the Digital Personal Identifier (DPI) that the user uses to identify him/herself to the service may be different in different contexts.

However, the role of user preferences does not end here. Once a service has been selected and customized, it will continue to execute until terminated. But while it is executing, conditions may change which cause the original selection to be no longer optimal in terms of user preferences. In Daidalos when a set of preferences is used to make a selection (service, device, network, etc.) or customize a service, the context conditions are stored. The Personalisation subsystem then registers with the Context Manager to be notified of any context changes that might affect these decisions. As long as the service continues to run, the system monitors the context of the user, and reacts to changes accordingly. Where this involves a change of service, the service is dynamically re-composed to change the preferred service.

By way of illustration, consider a university lecturer, Maria, going to give a lecture. She has with her a mobile phone and her PDA. When she enters the lecture room, the system automatically switches the ring tone of her phone to mute. Before entering the lecture room, she requests her slide presentation for the class. The appropriate service is selected, composed and started on her PDA. However, once inside the lecture room, as she approaches the front, the system discovers the large screen at the front of the lecture room and recomposes the service to use the large screen display. The first action is an example of changing the customisation in accordance with the preferences as the context of the user changes. The second is an example of reselection and re-composition of a service due to a change in context.

This form of proactivity is restricted in that it only applies to services initiated by the user. On the other hand it is sufficiently powerful to adapt the services either through parameters passed to the service affecting the way in which they are customised or through selection of an alternative service and re-composition to use this preferred service.

To assist the user in building up and maintaining their user preferences, several techniques are employed. These include:

(1) A user-friendly GUI is used to set up preferences manually.

(2) A set of stereotypes enables one to select an initial set of preferences.

(3) The user's actions are monitored and incremental learning employed to discover new preferences or refine existing ones.

(4) An offline data mining approach is used to analyse larger quantities of history data on user's actions and context.

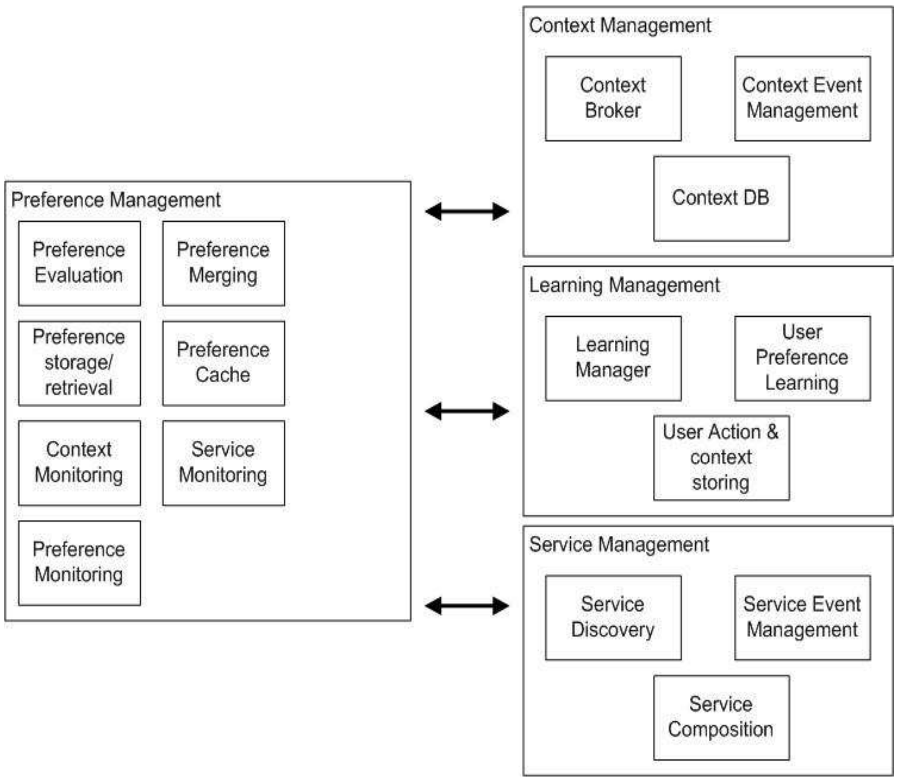

Fig. 3. Components of the Daidalos Pervasive Services Platform required for proactivity

As shown in Fig. 3, the Preference Management component includes the functionality that allows a preference to be created, stored, retrieved, merged with existing preferences, monitored and evaluated against current values of certain context attributes and current status of services. When a service is executed, any preference that affects the status of the service or a personalisable parameter of that service is an *active preference*. For each such active preference the Context Monitoring component registers for changes in specific context attributes affecting it. When a context event is fired from the Context Event Management, Context Monitoring is triggered to request the evaluation of these preferences by Preference Monitoring which cross-references the context condition with the preferences to locate all the active preferences affected by the change. The same process is performed by Service Monitoring which receives events from the Service Management component. If the status of a service changes, two tasks have to be performed. The first is to check whether this change affects any of the currently active preferences and order a re-evaluation of them. The second is to load or unload the conditions and preferences from the monitoring components if a service starts or stops accordingly and register or unregister for events that affect the service that started or stopped.

4.2 Proactivity in Persist

The Persist system is following a more general approach to proactivity, which extends that used in Daidalos. The most important aspect of this involves supplementing the Preference subsystem with a User Intent subsystem. These two separate subsystems drive the proactivity mechanism.

As far as the Preference subsystem is concerned, as in Daidalos the system is responsible for creating and maintaining the set of user preferences for a user. It does so by monitoring the actions taken by the user and the context in which they occur, and building up a history of (context, action) tuples. The snapshot of context selected consists of a number of context attributes, such as the user's location, the type and name of the service the user is currently interacting with, other services currently running in the PSS, etc. The final choice of attributes is critical to the success of both this and the User Intent subsystem. By applying learning algorithms to this the system can identify recurring situations and from this extract user preferences. These essentially have the form:

if <condition> do <action>

Thus if the system identifies through monitoring of the user's actions that the user always starts a particular service in a particular context, then it can set up a preference that will start the service for the user when that context arises. A preference might be reactive in that it responds to a user action as in the case of the previous example where whenever the user requests a news service, if the user is at home the system selects the television set. Alternatively it may be proactive, and initiate an action on the user's behalf when a particular context arises. For example, suppose that whenever the user arrives home after work in the winter, he/she switches up the temperature on the central heating. If the learning system identifies this, a preference rule could be set up to initiate this automatically for the user.

Unfortunately things are not quite as simple as this. When a recurring situation is identified, the action associated with the context condition may not always be performed. Equally, a preference may not have been used for a while and may have become out of date, eventually to be replaced by newer ones. To take account of these situations, the system maintains a set of attributes associated with each preference. These include a certainty factor representing the probability that this action is performed when the condition is satisfied, and a recency factor representing the date when this preference was last applied.

The second component responsible for driving the proactivity mechanism is the User Intent subsystem. This is similar to user preferences but in this case the aim is to look ahead and predict future actions based on past patterns of actions. The distinction between the two subsystems lies in the immediacy of the actions. User preferences are an immediate mechanism in that when a particular context condition arises, a corresponding action is taken. User intent is concerned with sequences of actions leading to a consequence at some later time in the future. Fig. 4 illustrates the main components of the PSS Framework for dealing with proactivity in a PSS.

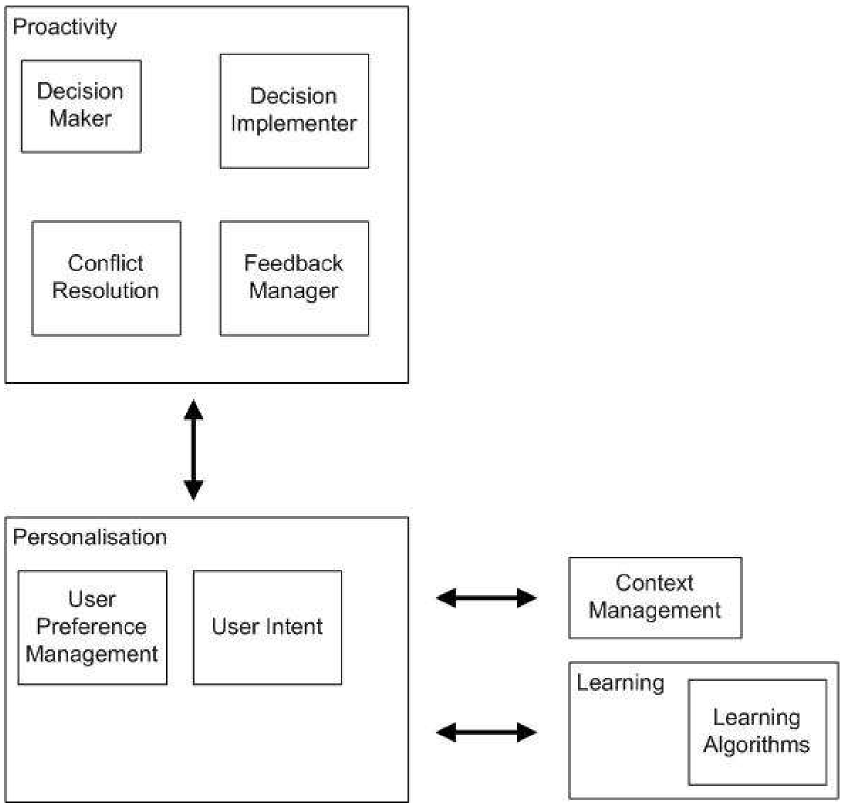

Fig. 4. Components of the PSS Framework for dealing with proactivity

To illustrate the idea behind user intent, consider the previous example of switching up the heating when the user arrives home after work in the winter. Suppose that the system recognises a pattern in the user's behaviour that starts with the user switching off his/her office computer at a time after 17.00. This may be followed by switching off the lights, exiting the building, starting the car, etc. If the system identifies that this pattern of actions always leads to the heating being switched on, it could store this as a user intent pattern. Thereafter whenever the system recognizes that the user is performing this sequence of actions, the resulting action can be carried out – in this case, it could send a message to trigger the user's heating system so that the house is at the required temperature for the user's arrival.

Hence the User Intent subsystem is responsible for the discovery and management of models of the user's behaviour. This is based on the actions performed by the user (in terms of interaction with services – starting a service, stopping a service, changing parameters that control the service, etc.), and tasks, where a task is a sequence of actions. In contrast to a user preference which specifies a single action to perform when a context situation is met, user intent may specify a sequence of actions to perform based on past and current user behaviour. This complements user preferences by predicting proactive actions in the future.

The User Intent subsystem consists of three main components:

(1) The User Intent Manager is responsible for overall control within the subsystem.

(2) The Discovery component is responsible for creating and maintaining the Task Model. This consists basically of the set of actions that make up each task and the probability of performing some particular action a after some other action b, or some particular task t after some other task u. Hence this captures the sequences of actions that a user might perform in the future based on past actions and contexts. To do so it needs to analyse the history of user actions and associated contexts and identify patterns in the sequences of actions that the user performs that can be classified as tasks. This history is the same set of (context, action) tuples used by the Preference subsystem. The tasks identified are represented in the Task Model. This construct is then passed to the Prediction component.

The Discovery component is invoked by the User Intent Manager from time to time to perform a new discovery cycle. The frequency with which this occurs is under the control of the system administrator. Typical factors that can be used to control this are:

- Time since last cycle.
- Number of actions executed since last cycle.
- Success rate judged by feedback mechanism.

The process used to create this Task Model is described in more detail in [14]. An illustration of the Task Model is given in Fig. 5

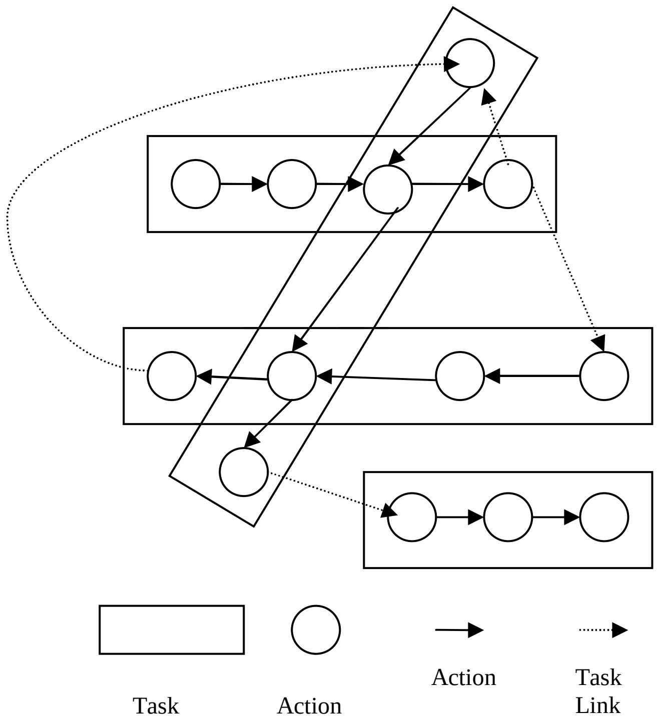

Fig. 5. Example of Task Model (from [14])

(3) The Prediction component is responsible for mapping the user's actions and associated context against the Task Model to identify potential tasks that the user might be engaged in, and once this is established with sufficient confidence, to infer any future action or actions that the user might be intending to perform. The result is passed to the Proactivity subsystem.

The Proactivity subsystem receives input from both the User Preference subsystem and the User Intent subsystem regarding the proactive actions recommended to be taken by the Proactivity subsystem. These inputs are passed to the Decision Maker component. As long as there is no conflict between these, the Decision Maker will trigger the Decision Implementer to initiate the recommended action. In doing so it also notifies the user and provides a means for the user to intervene if necessary and stop the action from proceeding. Whether the user does so or not, the Proactivity component informs the appropriate subsystem (Preference or User Intent) as to the user's acceptance or otherwise of the action concerned. This provides a useful feedback mechanism for updating the attributes associated with preferences and tasks. In the User Intent subsystem this is dealt with by the Feedback Manager.

However, the presence of two separate components (User Intent and User Preference Management) that can each provide predictions on future user behaviour, can lead to conflicting actions or actions that can cancel each other. Thus to assist the Proactivity subsystem in deciding how to respond to proactive action predictions, associated with each such prediction there is a degree of certainty. The Decision Maker uses this to ensure that the confidence of any prediction is high enough to implement. Where more than one action has been proposed, this can be used as a filter to remove any action for which the confidence level is too low.

The problem arises when there is a conflict between the actions requested by the two (User Intent and the User Preference Management components) which cannot be resolved by this simple filtering. This is referred to a separate component in the Proactivity subsystem responsible for Conflict Resolution. In such a case, the latter component uses a conflict resolution algorithm to determine what, if any, action should be taken. If the conflict resolution algorithm is unable to resolve the conflict, the user is prompted to provide an answer using the Feedback Manager. The user's input is then implemented and the User Intent and the User Preference Management components are informed of the success or failure of their predictions.

5 Conclusion and Future Work

This paper is concerned with the way in which proactivity may be handled in a pervasive system. It focuses on two separate pervasive systems:

- Daidalos: developed a pervasive system for mobile users based on an infrastructure that supports the provision of a wide range of personalized context aware services in a way that is easy for the end-user to manage and use.
- Persist: is developing a pervasive system based on the notion of a Personal Smart Space (PSS), which does not depend on access to complex infrastructure and can function independently wherever the user may be.

The Daidalos system uses a simple approach based on user preferences that can react to changes in the user's context to alter the customisation of a service or dynamically re-select and recompose a service if a change in user context needs this.

The Persist system extends this idea by including the ability to perform any user action. This includes the ability to start and stop services. This system also includes a user intent subsystem, based on a task model consisting of sequences of user actions and associated contexts. These two sources of proactive information are fed into the Proactivity subsystem, which determines when to perform an action on behalf of the user and what action is required.

A number of open issues still need to be tackled in this subsystem. These include the pattern discovery algorithm(s) to be used in task discovery. The issue of associating context with the actions and task discovered also needs further research. Work is also required on the format in which the discovered task model is stored as storing it as a graph with the associated context information rapidly becomes a problem for a mobile device with limited memory capabilities. Another possibility which could be of benefit is to use a priori knowledge of tasks, defined explicitly by the user or based on tasks performed by other users, to create or build up the Task Model. Such tasks could improve the accuracy of User Intent predictions at an early stage of system usage when there is insufficient history to create an accurate model.

Acknowledgments. This work was supported in part by the European Commission under the FP6 programme (Daidalos project) as well as under the FP7 programme (PERSIST project) which the authors gratefully acknowledge. The authors also wish to thank colleagues in the two projects without whom this paper would not have been possible. Apart from funding these two projects, the European Commission has no responsibility for the content of this paper.

References

1. Weiser, M.: The computer for the 21st century, Scientific American 265(3), 94-104 (1991).
2. Satyanarayanan, M.: Pervasive computing: vision and challenges, IEEE PCM 8(4), 10-17 (2001).
3. Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., Zhang, S.X.Q.: The Intelligent Home Testbed. In: Anatomy Control Software Workshop, 291-298 (1999).
4. Gopalratnam, K., Cook, D.J.: Online Sequential Prediction via Incremental Parsing: The Active LeZi Algorithm, IEEE Intelligent Systems 22(1), 52-58 (2007).
5. Román, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces, IEEE Pervasive Computing 1, 74–83 (2002).
6. Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: Experiences with ubiquitous computing rooms. IEEE Pervasive Computing 1, 67–74 (2002).
7. Kindberg, T., Barton, J.: A web-based nomadic computing system. Computer Networks 35, 443–456 (2001).
8. Sousa, J.P., Poladian, V., Garlan, D., Schmerl, B., Shaw, M.: Task-based Adaptation for Ubiquitous Computing, IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems 36(3), 328 – 340 (2006).
9. Kroner, A., Heckmann, D., Wahlster, W.: SPECTER: Building, Exploiting, and Sharing Augmented Memories. In: Workshop on Knowledge Sharing for Everyday Life. (KSEL06), (2006).
10. Si, H., Kawahara, Y., Morikawa, H., Aoyama, T.: A stochastic approach for creating context aware services based on context histories in smart Home. In: 1st International Workshop on Exploiting Context Histories in Smart Environments, 3rd Int Conf on Pervasive Computing (Pervasive 2005), 37-41 (2005).
11. Byun, H.E., Cheverst, K.: Utilising Context History to Provide Dynamic Adaptations. Journal of Applied AI, 18(6), 533-548 (2004).
12. Williams, M.H., Taylor, N.K., Roussaki, I. Robertson, P., Farshchian, B., Doolin, K.: Developing a Pervasive System for a Mobile Environment. In: eChallenges 2006 – Exploiting the Knowledge Economy, IOS Press, 1695 – 1702 (2006).
13. Crotty, M., Taylor, N., Williams, H., Frank, K., Roussaki, I., Roddy, M.: A Pervasive Environment Based on Personal Self-Improving Smart Spaces. Ambient Intelligence 2008, Springer Verlag, (2009).
14. Abu-Shabaan, Y., McBurney, S.M., Taylor, N.K., Williams, M.H., Kalatzis, N., Roussaki, I.: User Intent to Support Proactivity in a Pervasive System. In; PERSIST Workshop on Intelligent Pervasive Environments, AISB 09, Edinburgh, UK, 3-8 (2009).