Putting the 'Personal' into Personal Smart Spaces

Sarah M. Gallacher[1], Eliza Papadopoulou[1], Nick K. Taylor[1], M. Howard Williams[1]

[1] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK.
{s.gallacher, e.papadopoulou, n.k.taylor, m.h.williams}@hw.ac.uk

Abstract. Personalisation in a pervasive environment is challenging but essential. Without it pervasive systems will be unable to manage environments in ways that satisfactorily meet the needs of individual users. However, the dynamic aspects of pervasive environments provide new and interesting challenges for personalisation. This paper looks at how personalisation, based on user preferences and tasks, is implemented within a Personal Smart Space (PSS).

Keywords: context-awareness, personalisation, pervasive, preferences, tasks, learning.

1 Introduction

The rapid growth in the numbers of sensors, devices and networks coupled with the ubiquitous availability of services is resulting in complex environments that may soon become unmanageable for the average user. The aim of pervasive computing is to alleviate this problem and assist the user by providing an intelligent environment where services and resources are managed on their behalf with minimal user intervention. Personalisation is key to realising this goal.

The aim of personalisation is to adapt the behaviour of the system to meet the needs of individual users, causing it to behave differently for different users or for different contexts and available resources. For a mobile user this is particularly relevant since as he/she moves around, both his/her context and available resources will change. The system must track these changes and adapt its behaviour when necessary. To do this effectively it needs to acquire user specific information on user behaviours and preferences indicating how to adapt the system in different contexts.

The Persist project [1] aims to provide a pervasive experience through an architecture based on the notion of a Personal Smart Space (PSS) [2]. In contrast to common pervasive trends the PSS attempts to bridge the gap between mobile users and static intelligent spaces. Section 2 presents the Personal Smart Space concept and the PSS platform architecture. Sections 3 and 4 describe several key concepts implemented within the PSS personalisation system to overcome the interesting and challenging issues of providing personalisation in a pervasive domain.

2 Personal Smart Spaces

Pervasive computing research often centres on the development of pervasive systems within a local or a global domain. When pervasive technology is applied to a local domain we refer to this as a *smart space*. This is a bounded physical environment filled with adaptive devices (such as lights, window shutters, etc.) that can be automatically managed to meet the needs of individual users. Such systems hold great potential for independent living. In a global domain the goal is to provide mobile users with devices, networks and services to meet their needs wherever they may be. The location of the user plays a large part in the decision making process.

These two research tracks have tended to remain independent of one another, resulting in islands of pervasiveness separated by voids in which the support for pervasiveness is limited. The PSS approach integrates these two by unifying local, or fixed, smart spaces (associated with buildings) and global pervasiveness, mediated via mobile ad hoc networks (associated with users). A PSS is based on a personal area network constructed from a variety of devices ranging from static resources (e.g. printer) to smaller mobile and wearable devices to minute devices such as smart dust.

A PSS can be *mobile* in which case its physical boundary moves with the PSS owner (who may be a person or legal entity) or *fixed* in that it is implemented within a static structure. Utilising ad-hoc network technology a PSS can interoperate with other PSSs for information and service sharing as shown in Fig. 1.

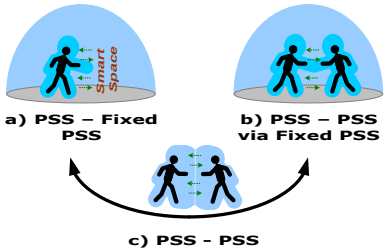

Fig. 1. Interaction between PSSs. a) a mobile PSS interacts with a fixed PSS; b) two mobile PSSs interact via a fixed PSS; c) two mobile PSSs interact without a fixed PSS.

2.1 The PSS Platform

For a device to be PSS-enabled it must run an instance of the PSS platform software. The platform provides various core PSS management functions (e.g. personalisation and proactivity) as well as components that support the *self-improving* aspect of the PSS (such as learning). Fig. 2 shows the architecture of the PSS platform.

Networking, messaging and service runtime are handled in the lower layers while the PSS Framework layer relates to user experience. One important aim of a PSS is self-improvement by learning to identify trends in user behaviour enabling recommendations and predictions for environment adaptations that can be performed automatically on behalf of the user. Several component blocks within the framework layer are key to meeting this goal:

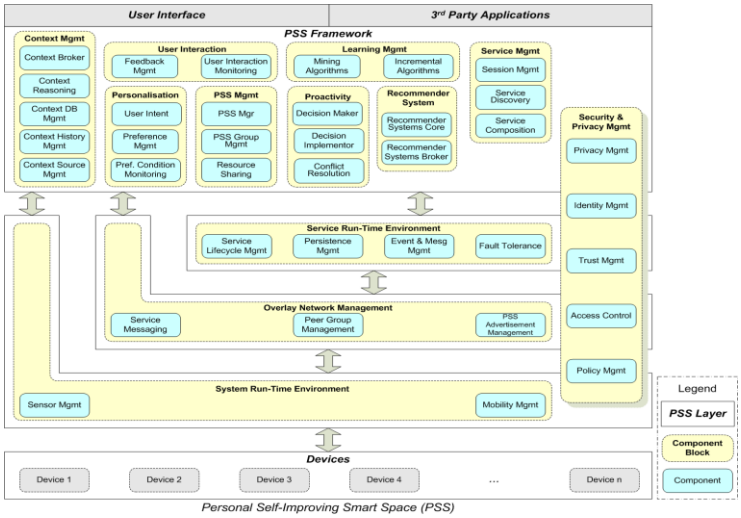

Fig. 2. PSS Platform Architecture

(1) Context Management (CM). This component block stores and manages the contextual information related to the PSS owner. Such information can originate from a variety of sources such as sensors or services (both enabling services within the platform and third party services) or can be entered explicitly by the owner. Context information is used throughout the platform.
(2) Personalisation. This block maintains and processes the stores of user preferences and user task models (stored in the CM database). It responds to requests for such preference information (through Preference Management) and user task information (through User Intent) as well as automatically re-evaluating such information as context changes occur.
(3) Learning Management (LM). The LM is implemented as a library store containing various learning algorithms (both batch mining and incremental) used for a variety of learning tasks throughout the framework.
(4) User Interaction. Within this block the User Interaction Monitor (UIM) captures user behaviours as the user interacts with services, networks and devices within their environment (e.g. selecting a service, turning off a light, etc.). Feedback Management provides GUIs used at the point of pro-active personalisation application.
(5) Proactivity. The Proactivity block processes several input streams that forward personalisation proposals (from Personalisation and Learning Management components). It decides what personalisation proposals to implement and resolves conflicts between proposals from different input streams if they arise.

The first version of the integrated platform is now available to download from the Sourceforge website [3]. Development still continues to enhance support for third party services with a view to fully demonstrating the platform in September 2010.

A qualitative analysis of the platform architecture was carried out with three separate user groups consisting of both males and females between the ages of 18 and 64. The groups were shown two scenarios illustrating how the PSS could provide benefit in terms of proactively personalising their environment based on learnt

behaviours and information exchange. Encouragingly the majority of users indicated they would be happy to use such a system and saw its potential to support them in organisational and management tasks. In addition a quantitative analysis has been outlined including a set of benchmark metrics that components must achieve. We are currently testing the delivered integrated platform against such metrics.

3 Dynamic Context-Dependent User Profile

Satyanarayanan [4] argues that in pervasive environments, higher-level user information is essential for system decision taking, since without such information it is impossible to know what adaptations will help rather than hinder the user. The store of such user information is often referred to as a *user profile* and acts as a driver for personalisation processes. Several profiling standards exist defining the content of user profiles. The CC/PP standard [5] relates to a web environment while the ETSI standard [6] relates to a mobile environment. These standards are useful in their respective domains but they do not support conditional user preferences.

Personalisation in a pervasive environment must be able to handle a level of dynamism not relevant to personalisation in other domains, such as web environments. This dynamic aspect is introduced through changing context. Firstly, the available resources will change as the user's PSS interacts with other PSSs (both fixed and mobile) in their vicinity. The availability of resources affects the possible functions that the user can perform. Secondly, users may have different requirements in different contexts. For example, a user might want to share a WiFi service when he is at work but not when he is in the shopping mall. In other words, the profile must include *context-dependent* user information that changes with the user's context.

The user profile within the PSS contains two types of context-dependent user information: *user preferences* and *user tasks*. Both are used in decision taking to achieve personalised adaptation. The latter differs from context-aware adaptation in that it can adapt the behaviour of the system differently for different users rather than adapting it uniformly for all users based on different situations.

3.1 User Preferences

The preferences of the user may be captured and represented in a variety of forms. In the Persist platform three different forms are used:
(1) Preference rules. These are user specific rules that dictate the needs of a user in some context. With this information the system can adapt its behaviour on behalf of the user when the particular contextual situation arises. Such rules may be displayed in human-readable form, affording users the possibility of viewing and manipulating them if so desired. This enables the user to retain a sense of control as well as a better understanding of what information the system holds and therefore what behaviours to expect. Such understanding is crucial for system acceptance (e.g. Dey [7]).
(2) Neural nets. Here the context and actions of the user are used to train a neural network. The preferences are stored as a set of weights associated with the nodes and

links of the network. Thus the user cannot view the result in any meaningful way although for some preferences it provides a more powerful mechanism.
(3) Bayesian networks. These are similar to neural networks in the way in which they are trained and used.

While all three formats are used in the Persist platform, this paper focuses only on the rule-based form as complete coverage of all three would make the paper too long.

Within a PSS, user preference rules are implemented as IF-THEN-ELSE rules. At a simple level a preference consists of two key parts, a condition and an outcome, e.g.

IF location=work AND status=free (condition)
THEN share_wifi=true (outcome)

The condition represents some contextual state and the outcome represents the action to perform depending on the truth of the condition. Note that the task of providing context tags such as symbolic location names and user status is a challenging issue. This is handled within the Persist platform by context inference techniques using an ontology developed for the purpose. However, this is outside the scope of this paper.

Since the definition of a preference rule is recursive, at a more complex level the outcome of one preference rule may be another IF-THEN-ELSE preference. Thus the structure of a preference rule has the potential to become large and complex. For this reason house-keeping operations are being developed to prune over-complex rules.

Since user preferences depend on context, when the context changes, the personalization may need to be re-evaluated. To handle this the PSS implements a preference condition monitor that listens for relevant context changes, re-evaluating the dependent preferences when a change occurs. If this results in a new outcome, Preference Management forwards this to Proactivity for implementation.

3.2 User Tasks

In addition to having context-dependent needs and preferences, a user may also exhibit recurring patterns of behaviour. This is often the case when the user performs a recurring task. For example, each time the user leaves work in the evenings they may perform the same sequence of actions: *turn off office lights, unlock car, set radio to news channel*. With this information the system can identify what task the user is currently performing and adapt the environment appropriately on their behalf to help them achieve their goals. Within the PSS tasks are implemented as hierarchical models similar to Hierarchical Hidden Markov Models (HHMM) [8]. At a simple level each task consists of a sequence of actions.

As with preferences each task may be recursive, where an action can be another task. Unlike preferences, the implementation of an action within a task is primarily dependent on the completion of previous actions within the task. However, context information is crucial to ensure that an action is implemented appropriately. It may be the case that some actions within a task cannot, or should not, be instigated until a particular context situation holds. For example, consider the following:

```
<task name=task_1>
    <action name=action_1>
        <condition>location=office</condition>
        <outcome>office_lights=off</outcome>
```

```
        </action>
        < action name=action_2>
                <condition>location=car</condition>
                <outcome>car_lock=unlocked</outcome>
        </action>
        …
</task>
```

In this example it would be undesirable for the car to be unlocked before the user is in the vicinity. Therefore, each action can have an associated context condition that must hold true before the action can be initiated.

Before a task can be implemented, the system must identify which task (if any) the user is trying to achieve. To do this, current user behaviour is monitored via the UIM. The system cross-references this with a task model to identify if a known task is being performed. If a task is identified User Intent forwards future actions to Proactivity for implementation.

3.2 Proactive Behaviours based on Profile Information

Since the user profile within a PSS consists of two types of user information (preferences and tasks), mechanisms are required to mediate both types for the resolution and implementation of automatic system behaviours. The Proactivity component (to be fully described in a future paper) provides this functionality.

It regularly receives inputs in the form of preference outcomes (from Preference Management) and predicted task actions (from User Intent). When either input type is received it is checked against the other input stream to identify possible conflicts. If a conflict is detected, confidence levels and quality of context values are used to find a resolution without involving the user (where possible). If no conflict is detected the input is implemented accordingly.

In both situations Feedback Management can be used to involve the user. The PSS provides several levels of feedback involvement allowing the user to be prompted according to system confidence. This ranges from explicit prompts (requiring user input) to implicit prompts (only requiring input if the user wishes to abort the operation) to automatic behaviour with no prompting. The aim is to reduce user interactions without completely removing the user from decision making processes.

4 Implicit Learning and Profile Management

For effective personalisation it is essential to maintain an up-to-date user profile. Some systems (e.g. [9, 10]) rely on the user to do this manually. However, this places a great burden on the user, often out-weighing the benefits personalisation aims to provide. Consider the following scenarios:

(1) A new user starts to use a PSS. Their initial profile will be sparse, if not empty. An extensive profile specific to the individual user must be created.

(2) An existing user starts to use a new service. Their profile may lack information to personalise the new service. New information may be required or existing information may need to be refined.
(3) An existing user changes their behaviour. Their profile must be updated to reflect the behaviour change.

The first scenario is inevitable for each new PSS user. The second and third scenarios will occur regularly throughout the lifetime of the PSS. This illustrates that profile management is an ongoing and regular task that could easily overwhelm the average user. Hence many recent pervasive projects (e.g. [11, 12]) have adopted *implicit* techniques utilising monitoring mechanisms (to gather historic user behaviour data) and machine learning algorithms (to extract useful information from such data).

The PSS monitors user behaviour (based on user interactions with services, networks and devices) via the UIM. The UIM performs passive monitoring; this means that the onus is on the resource the user is interacting with to send the UIM notifications of user actions. This passes control back to the monitored resource and removes static dependencies on roaming resources. When the UIM receives a notification of an interaction, context history mechanisms store the interaction with a set of relevant context attributes, describing the context in which the interaction occurred. This set of historic behaviour data may then be processed by various machine learning algorithms to extract user preferences and user tasks.

For the purposes of the Persist platform, where the study of different learning mechanisms was not one of the objectives, one mechanism was selected for demonstration; the C4.5 decision tree algorithm [13] (an extension of Quinlan's ID3 algorithm). Each batch execution of the algorithm generates a set of decision trees, representing a preference. This is then translated into the internal preference format (shown above) and merged with the user's existing preference set within their profile. This rule based approach is used in several projects [14, 15] and has been successfully implemented and evaluated in the Daidalos project platform (to appear in forthcoming paper). Although a lack of history can often lead to inaccurate learning in the early stages [16], negative impact can be mitigated through the use of appropriate feedback and prompting systems. In the PSS such functionality is provided by the Feedback Management component.

Task discovery is performed in two stages. Firstly, repeated sequences of actions are discovered using pattern discovery techniques. This process generates a list of common action sequences which are translated into the task format (shown above) and stored as the user's task model. Other projects such as MavHome [17] and Synapse [18] use similar methods for task discovery; however, Persist incorporates context-dependencies into tasks to mitigate incorrect or premature behaviours. Each task is analysed to identify actions that only occur in specific context situations. This specific context information is added to the relevant actions as a context condition indicating that the action should only be initiated if the context condition holds.

Acknowledgements. This work is supported by the European Union under the FP7 programme (Persist project) which the authors gratefully acknowledge. The authors wish to thank all colleagues in the Persist project developing Personal Self-Improving Smart Spaces. However, it should be noted that this paper expresses the authors' personal views, which are not necessarily those of the Persist consortium. Apart from

funding the Persist project, the European Commission has no responsibility for the content of this paper.

References

1. Persist project homepage: http://www.ict-persist.eu, accessed on 16th April 2010.
2. Taylor, N.K.: Personal eSpace and Personal Smart Spaces. In: First PerAda Workshop on Pervasive Adaptation, (SASO'08), Venice, Italy (2008)
3. Personal Smart Space open source: http://sourceforge.net/projects/psmartspace, accessed on 16th April 2010.
4. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications 8(4), (2001) 10 - 17.
5. W3C: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0. Available online: http://www.w3c.org/TR/2007/WD-CCPP-struct-vocab2-20070430/, accessed on June 17th 2009. (2007)
6. ETSI: Human Factors (HF); User Profile Management, ETSI Guide, EG 202 325 v1.1.1. Available online: http://pda.etsi.org/exchangefolder/eg_202325v010101p.pdf, Accessed on June 17th 2009. (2005)
7. Dey, A.K.: Modeling and Intelligibility in ambient environments. Journal of Ambient Intelligence and Smart Environments 1(1), (2009) 57 - 62.
8. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden Markov model: Analysis and applications. Machine Learning 32(1), (1998) 41 - 62.
9. Yoshihama, S., Chou, P., Wong D.: Managing Behaviour of Intelligent Environments. In: Proceedings of PerCom '03, (2003) 330 - 337.
10. Sousa, J.P., Poladian, V., Garlan, D., Schmerl, B., Shaw, M.: Task-based Adaptation for Ubiquitous Computing. In: IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems, 36(3) (2006) 328 - 340.
11. Ziebart, B.D. Roth, D., Campbell, R.H., Dey, A.K.: Learning Automation Policies for Pervasive Computing Environments. In: Proceedings of the 2nd International Conference on Autonomic Computing (ICAC '05), (2005) 193- 203.
12. Groppe, J.: Profile Management Technology for Smart Customisations in Private Home Applications. In: Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA '05). (2005)
13. Quinlan, J.R.: C45: Programs for Machine Learning. Morgan Kaufman. (1993)
14. Cordier C.: Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform. Workshop on Applications and Services in Wireless Networks (ASWN '06). (2006)
15. Strutterer, M.: Managing and Delivering Context-Dependent User Preferences in Ubiquitous Computing Environments. Proc. of 2007 International Symposium on Applications and Internet Workshops (SAINTW '07). (2007)
16. Webb, G.I., Pazzani, M.J., Billsus, D.: Machine Learning for User Modeling. User Modeling and User-Adapted Interaction 11(1-2), (2001) 19 - 29.
17. Youngblood, M.G., Holder L.B., Cook, D.J.: Managing Adaptive Versatile Environments. Proc. of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom '05), (2005) 351 - 360.
18. Si., H.K.Y., Morikawa, H., Aoyama, T.: A Stochastic Approach for Creating Context-Aware Services on Context Histories in Smart Home. In Proceedings of ECHISE2005, Pervasive 2005, (2005) 37 - 41.