Proceedings of the Symposium

PERSIST Workshop on Intelligent Pervasive Environments

A symposium at the AISB 2009 Convention (6-9 April 2009)
Heriot-Watt University, Edinburgh, Scotland

Symposium Chairs
Sarah M McBurney
Elizabeth Papadopoulou

Published by SSAISB:
The Society for the Study of Artificial Intelligence
and the Simulation of Behaviour

http://www.aisb.org.uk/

ISBN - 1902956834

PERSIST Workshop on Intelligent Pervasive Environments

A one-day symposium at AISB 2009 (6-9 April 2009).
http://www.macs.hw.ac.uk/~ceeep1/PersistWorkshopAISB09/index.php

PROGRAMME CHAIRS

Eliza Papadopoulou, Heriot-Watt University, UK
Sarah McBurney, Heriot-Watt University, UK

INTRODUCTION

Pervasive environments represent a vision of the future where environments populated with computational technology adapt to meet the needs of individual users. Networks, services and devices should be ubiquitously available and work in concert to support each individual in carrying out their everyday life activities, tasks and rituals in transparent and unobtrusive ways. To meet this challenge, environments and the devices within should not only be adaptable but also intelligent to provide real benefit rather than hindrance to the user.

Firstly, there is the challenge of adaptable network, service and device provision. How can such entities be ubiquitously available and adapted to meet user needs? The discovery, management and configuration of pervasive entities should be performed on behalf of the user. These processes should be continuous as the user moves through their pervasive environment. Intelligence is essential in such processes to allow for an environment which meets the user's needs with minimal user input.

Secondly, how should a pervasive system facilitate the management and communication of system intelligence within a pervasive environment? Context awareness and personalisation processes provide invaluable knowledge for system decision taking. However, how should such knowledge be gathered? How is it shared and exploited most beneficially and how could higher level knowledge be accurately inferred? Explicitly asking the user to provide and manage such information is undesirable and goes against the pervasive ideology.

Finally, how should system intelligence and information be kept private and secure? Pervasive systems could potentially amass large stores of sensitive user information. How can such information be securely communicated throughout the system as well as to third parties. Further, how should the user's identity be kept private and when should their associated data be shared or concealed?

The PERSIST project is a Framework 7 project that began in April 2008 to address these challenging issues with the notion of a Personal Smart Space. The Consortium consists of 10 industrial and academic partners from across Europe. For more information on the PERSIST project please see the website at:
http://www.ict-persist.eu/

TOPICS

Topics of interest include but are not limited to:

- Objects, devices and environments that embody intelligent pervasiveness
- Context awareness, sensing and inference
- Distributed software, systems, middleware and frameworks
- Analysis, design, implementation and evaluation
- Security, privacy and trust
- Communication systems and infrastructure
- Service discovery and management
- Personalisation and user modelling

Accepted papers cover a wide range of these topics.

ACKNOWLEDGEMENTS

The symposium organization would not have been possible without the dedicated support of the internal Programme Committee as well as the support of the AISB 2009 Convention Chair Dr. Nick Taylor and his Organisation Committee.

PROGRAMME COMMITTEE

Yussuf Abu-Shaaban, Heriot-Watt University, UK
Micheal Crotty, Waterford Institute of Technology, Ireland
Kevin Doolin, Waterford Institute of Technology, Ireland
Pierfranco Ferronato, Soluta.net, Italy
Korbinian Frank, German Aerospace Centre, Germany
Jan Porekar, Jozef Stefan Institute, Slovenia
Ioanna Roussaki, National Technical University of Athens, Greece
Charles Sheridan, Intel, Ireland
Nick Taylor, Heriot-Watt University, UK
Claudio Venezia, Telecom Italia, Italy
John Whitmore, Lake Communications, Ireland
Howard Williams, Heriot-Watt University, UK

Table of Contents

User Intent to Support Proactivity in a Pervasive System

Yussuf Abu Shaaban[1], Sarah McBurney[1], Nick Taylor[1], M. Howard Williams[1], Nikos Kalatzis[2] and Ioanna Roussaki[2]

Abstract. In a pervasive system it is essential to understand the intent of the user in order to predict his/her future behaviour. This in turn will help to minimise the user's administrative overheads and assist the user to achieve his/her goals. The aim of this paper is to present some aspects of how user intent may be handled. It focuses on the architecture supporting the proactive features of the Persist pervasive platform. A formal definition of the task discovery problem in user intent is provided. The use of the discovered task model to predict the user's next intended task/action is introduced including the way in which user context can assist in the prediction of the user's intended task/action.

1 INTRODUCTION

In a pervasive environment with ubiquitous access to services, networks and devices it is essential that mechanisms are in place to mitigate the user's resource management responsibilities and aid the user in daily tasks. Such mechanisms should be based on high level knowledge of the user's preferences and intentions, and the resulting user behaviour. Without such knowledge it is difficult for a pervasive system to identify accurately what actions will help rather than hinder the user.

The Daidalos project developed a pervasive system which included a personalisation and preference management subsystem (including learning) which implicitly gathered and managed 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 personalisation subsystem allowed the system to personalise the user's environment in an unobtrusive and beneficial way (based on previous user behaviour). However, this personalisation mechanism was solely based on current context and therefore its ability to predict future actions was limited. For example, if the user always turns on the heat when they return home, preferences cannot trigger such an action on behalf of the user until the user is in the home context.

The Persist project is an FP7 EU project which started in April 2008. It aims to create a rather different form of pervasive system but in doing so it will extend and adapt some of the developments of the Daidalos system. In particular, it will complement the personalisation and preference management system with a user intent system. The aim of the user intent system is to discover and manage a model of the user's behaviour in the form of tasks and actions. An action can be any interaction between user and a service while a task is a sequence of actions. Whereas a user preference specifies one action to perform when a context situation is met, user intent will specify a sequence of actions to perform based on past and current user behaviour. This overcomes the limitations on forecasting future behaviour and preferences enabling the prediction of environment adaptations in the future.

Returning to the earlier example, user intent may recognise a 'going home' task which starts when the user switches off their computer and office lights. When the system identifies that this task is being performed, it could trigger the user's heating system so that the house is at the required temperature for the user's arrival.

Both user intent predictions and preferences will provide input to proactivity mechanisms within the Persist framework. With the addition of user intent predictions, proactive mechanisms can perform operations well in advance providing an environment that minimises user involvement and enhances user experience.

The rest of the paper is structured as follows. The next section looks at related work investigating user-intent for proactivity in pervasive systems. Section 3 introduces the notion of a Personal Smart Space (PSS) and describes the high level design of the Persist architecture. Section 4 illustrates the architecture of the Persist User Intent system. Section 5 concludes and details future work.

2 RELATED WORK

In the past various projects have addressed the problem of adapting environments in a proactive manner. Among the pioneers were IBM's Blue Space [1] and UMA's Intelligent Home project [2], which based proactive adaptations on user preferences. However users had to manually create and maintain their preference set. This is no trivial task and the burden of such information management responsibilities led to a sparse preference set. Therefore, only basic personalised environment adaptation was provided by these systems. Another project that addressed this challenge was Aura [3]. Aura attempted to incorporate user intent to aid proactive actions. However as with the previous projects the user was expected to manually enter high level information, such as the user's current task, as well as basic preference information. Once again this approach proved to be inefficient, as the burden on the user was not mitigated.

The MavHome[4] project attempted to reduce the user's information management responsibilities by facilitating monitoring and learning mechanisms to gather user information unobtrusively. In more detail, MavHome aims to provide a house with mechanisms capable of maximizing inhabitants' comfort and minimizing operational cost by predicting the user's intentions with regard to mobility patterns and device usage. In order to achieve this, MavHome models locations inside the house by creating a dictionary of zone identities treated as character symbols and gathers statistics based on the history of user movement contexts, or phrases. The prediction algorithm used is called "LeZi-update" and is based on the dictionary-based LZ78 compression algorithm. In order to predict the user's next action, the system identifies patterns observed in past

[1] Department of Computer Science, School of Mathematical and Computer Sciences, Heriot Watt University, UK. Email: {ya37, ceesmm1, nkt, mhw}@macs.hw.ac.uk.

[2] School of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Athens, Greece. Email: {nikosk, Ioanna.Roussaki}@cn.ntua.gr.

inhabitants' activities. User actions are represented by characters, which are monitored and stored in a history log. The algorithm used is called Smart Home Inhabitant Prediction (SHIP) that basically matches the most recent sequence of events with sequences in collected histories.

Specter [5] is a mobile personal assistant aiming to assist users in their everyday life tasks or situations. The system learns and binds situations and services between the user and the system, in a collaborative process. Specter proposes a memory model that consists of two main parts that cater for short-term memory and long-term memory. Contextual data provided by the environment is initially collected and maintained in short-term memory and forms a snapshot of current user context. At this point, context-aware services can be provided via rules which have been previously defined by the user. When stored information becomes outdated, it is transferred to long-term memory, where there are opportunities for review and evaluation by the user in a process called introspection. The process performed on long-term memory leads to a learned user model that reflects the user's situation and activities.

Synapse [6] is a context-aware service platform for the provision of specific services to users. Synapse learns different patterns of user behaviour (i.e. tasks) by exploiting the recorded histories of context and services. Based on the user's current situation and task, Synapse can predict and provide the most appropriate services. The creation of a user model is based on Bayesian networks or on Hidden Markov Models. In case there is a system uncertainty, the system asks for user input, something that allows for more accurate personalization, but at the same time the user might be disturbed by too many pop-up messages. In [7], a system is presented that supports proactive, modelling-based, adaptations in a user's office. The system learns the patterns of the user's behaviour in an office environment by utilizing context history. Initially, the user controls the environment of an office (e.g. opening/closing windows, turning lights on/off, and adjusting the temperature). Gradually, the system learns the user's situation and behaviour as the size of context history data increases. After some time, the system is capable of providing dynamic adjustments based on the created user model, without the need for predefined rules. The system includes two databases, one for storing context history and one for storing the learned user model. The learning process is based on a fuzzy set based decision tree learning algorithm.

3 THE PERSIST SYSTEM

A Personal Smart Space (PSS) is defined by a set of services within a dynamic space of connectable devices where the set of services are owned, controlled, or administered by a single user or organisation. It facilitates interactions with other PSSs, is self-improving and capable of proactive behaviour. Thus, a PSS is: user centric, always controlled by a single user; it is mobile; it allows interactions with other PSSs and is capable of self-improvement.

This section elaborates on the Persist system and more specifically, on the PSS high level architecture design. In order to provide a high level specification of the Persist architecture, five main layers have been distinguished in the functional design, where each layer incorporates various component blocks and components that are essential to the design of the PSS environment. The layered PSS architecture of Persist is depicted in Figure 1.Each layer addresses a well defined part of the PSS functionality. The names and purpose of these layers are presented hereafter.

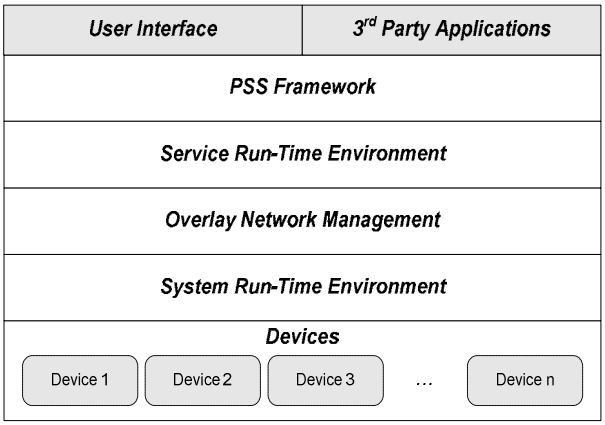

Figure 1. The high level architecture of Personal Smart Spaces

Layer 1 - Devices

The PSS definition suggests that a single PSS can span many different devices. Depending on their processing and networking capabilities, these devices may either implement the PSS stack or part of it, or simply interact with the rest of the PSS framework.

Based on a study of the functional/technical commonalities of current communication and computing leading technologies, the devices that may be part of a PSS have been classified as follows: (i) *servers* (i.e. independent computers dedicated to provide one or more services over a computer network; e.g. Windows Media Center, Apple Itheatre, PCs), (ii) *laptops* (i.e. small-sized portable computers; e.g. Mac Book, Sony Vaio, Tablet PC), (iii) *mobile phones* (i.e. pocket-sized handheld computing devices; e.g. iPhone, HTC Tytan, Nokia N90, PDA), (iv) *sensors* (i.e. group of devices that may be embedded into other devices, can measure a physical parameter and convert it into a signal, which in turn can be read by an observer or an instrument; e.g. RFID readers, GPS location estimators, accelerometers, thermometers, altimeter, barometer, air speed indicator, signal strength measurer), (v) *smart objects* (i.e. resource-constrained devices that can be connected to the Internet or a LAN via a wifi connection, ethernet, GPRS, 3G, etc., usually intended for displaying multimedia content such as a combination of text, audio, still images, animation and video or other everyday objects enhanced with pervasive facilities; e.g. WiFi photoframes, Chumby, Nabaztag, home eAppliances, surveillance cameras) and (vi) *interactive entertainment electronic devices* (i.e. interactive entertainment electronic devices producing a video display signal, which can be used with a display device (a television, monitor, etc.) to display a video game or an external source of signal, such as ipTV; e.g. set-top box, gaming console).

Layer 2 - System Run-Time Environment

The System Run-Time layer of the PSS architecture serves as an abstraction layer between the underlying device operating

system and the PSS software, in order to achieve a high degree of platform independence. Essentially, this layer is the one that makes a device PSS-enabled. Hence, employing an "off-the-shelf" implementation of a virtual machine run-time will offer PSS portability over a wide range of software and hardware platforms. This layer is also responsible for the device mobility and sensor management.

Layer 3 - Overlay Network Management

The Overlay Network Management layer provides the PSS architecture with a Peer-to-Peer (P2P) management and communication layer. The services within this layer provide functionality for PSS peer group management, PSS peer discovery, peer PSS group discovery, PSS communication management and message routing between peer networks of PSSs. It is assumed that the lower level ad-hoc networking functionality will be managed by other 3rd party components and therefore it is considered outside the scope of the Persist Project.

Layer 4 - Service Run-Time Environment

The Service Run-Time Environment layer provides a container for the PSS services. It supports service life cycle management features and provides a service registry, as well as, a device registry. Moreover, it allows for service management in a distributed fashion across multiple devices within the same PSS. In this context, it delivers fault tolerance as well as device resource management. The Service Run-Time Environment also provides advanced information management features for achieving high availability of data, for addressing storage requirements of PSS services, and for supporting event and message management.

Layer 5 - PSS Framework

The PSS Framework layer is the core of the PSS architecture. Its functionality includes service discovery, composition and session management (both PSS and 3rd party services) as well as management of context information, including user preferences. Moreover, the PSS Framework layer supports inference of context information, automatic learning of preferences, and identification of user's future intentions. This information, together with data provided by the recommender system of this layer, enables the proactive facilities of the PSS platform. The PSS Framework layer also offers support for user interaction monitoring as well as user feedback collection and management. Furthermore, this layer provides support for conflict resolution, grouping of context data and preferences and resource sharing. Finally, the PSS Framework layer enables security and privacy management, demonstrating features such as access control, identity management, privacy and trust management, and policy management. However, it should be mentioned that some security and privacy facilities of the PSS also need support from layers 2, 3 and 4 to enable a fully secure and privacy-aware PSS system.

4 OVERVIEW OF THE USER INTENT SUBSYSTEM

As described in Section 3, to support the proactive behaviour in Persist, user intentions need to be predicted. Figure 2 illustrates the User Intent subsystem. Controlled by the User Intent Manager, the Discovery component is responsible for identifying a set of tasks the user could perform in the future in specific contexts based on learning from the user's history of actions and contexts. The Task Model produced, composed of the tasks identified, is passed to the Prediction component. Based on actions the user recently performed and the user's current context, the Prediction component uses the Task Model to infer the next action/task the user is intending to perform. This prediction is passed to the Proactivity component which later sends feedback to User Intent regarding the correctness of the task/action predicted, as judged by the user's reaction to the action taken.

The rest of this section is structured as follows. An overview of task discovery is provided in Section 4.1. This is followed in Section 4.2 by a description of task/action prediction to support proactivity.

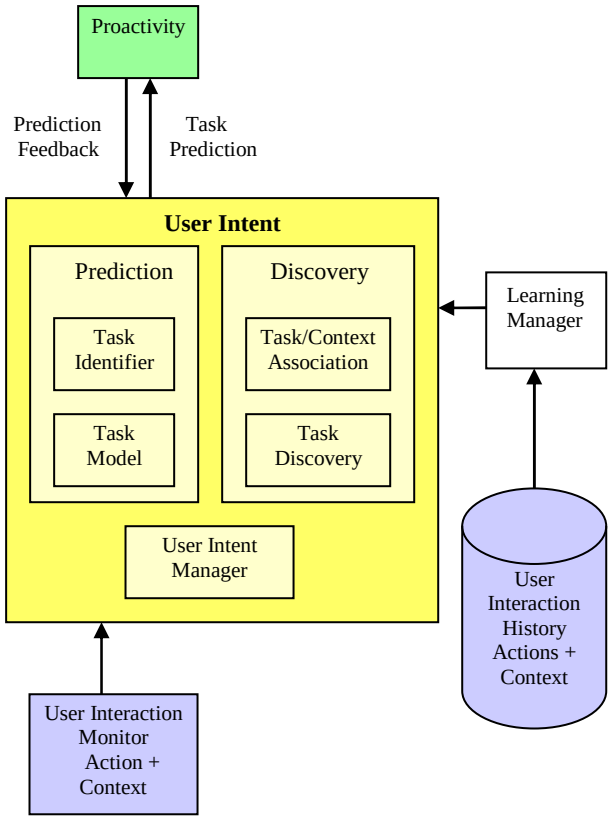

Figure 2 The User Intent System Architecture

4.1 Task and Context Pattern Discovery

The Discovery part of User Intent shown in Figure 2 identifies a set of tasks the user might perform in the future based on user history. Formally, user history can be expressed as a set $\{(a_1, c_1), (a_2, c_2),\ldots,(a_n, c_n)\}$ where (a_i, c_i) represents the action a_i performed by the user in context snapshot c_i. A number of attributes can be included in a context snapshot such as location,

the type and name of the service the user is currently interacting with, other services currently running in the PSS, etc.

The task discovery problem includes recognising a set of tasks T_1, T_2, …,T_s, where T_j is composed of a sequence of actions as illustrated in Figure 3. $P_{t\{i,j\}}$ denotes the probability of starting T_j on completion of T_i. The probability of performing a_y after a_x is $P_{a\{x,y\}}$ It is possible for one action to be part of more than one task.

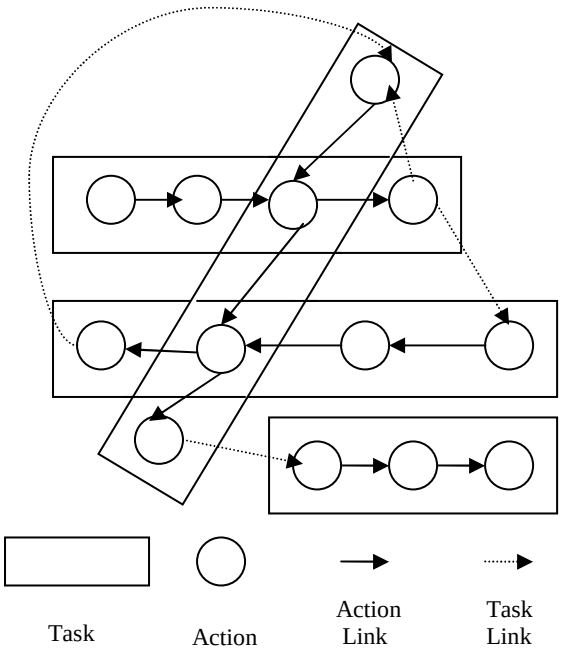

Figure 3 Task Model Discovery

As illustrated in Figure 4, the User Intent Manager triggers a new discovery cycle based on the following factors:

- The number of prediction hits as indicated by the Proactivity prediction feedback.
- The number of actions executed since the last discovery cycle.
- The time elapsed since the last discovery cycle.

The following steps are performed in a discovery cycle:

- The User Intent Manager instructs Task Discovery to start the discovery cycle.
- On Task Discovery request, the Learning Manager applies a pattern recognition algorithm to the user history in order to detect patterns of user actions. Each pattern identified is a potential user task. The probability of performing a task given the previous task performed is computed. The probability of undertaking an action given the previous action performed is also computed.
- Once the tasks are identified, associating user context to the actions forming the tasks discovered is required, as the user could in the past have performed the same task in different contexts. Analysis is performed by the Task/Context Association component to associate a unified context with the actions discovered.
- The Task Model is updated with the new task discovery model.

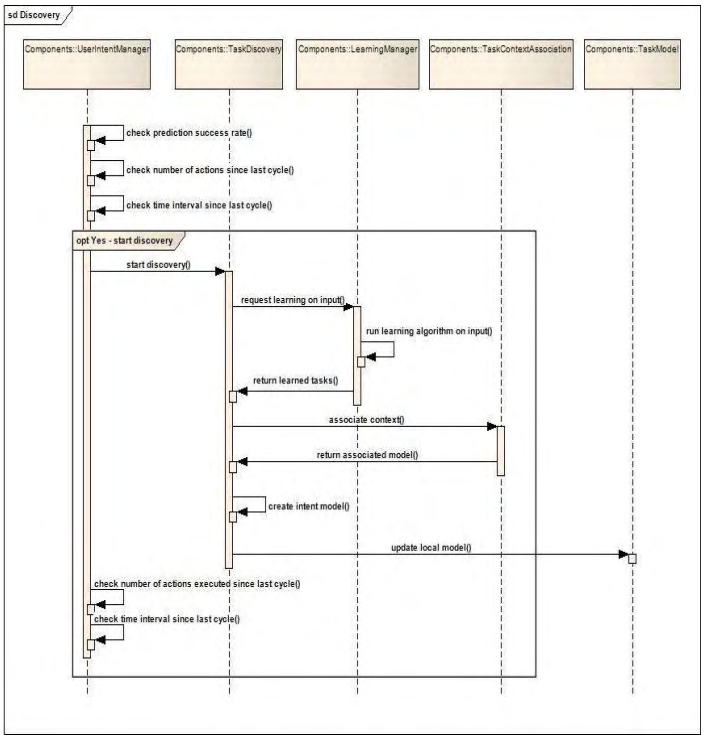

Figure 4 The Task Discovery Cycle

4.2 Task Prediction

The Prediction part of User Intent uses the Task Model produced by Discovery to predict the next task/action the user is intending to perform. A number of factors could be taken into account in predicting the next intended task/action which includes: the previous action predicted, the feedback from Proactivity on the accuracy of this prediction, the action the user actually performed (if different from what was predicted) and the list of subsequent correct predictions made previously. An initial proposal for an algorithm to predict user intention is given below. A prediction cycle starts when the User Intent Manager receives from User Interaction Monitoring the details of the current action the user is performing. The Task Identifier checks the current task prediction against the action that the user has performed. Figure 4 illustrates the prediction steps required when the current predicted task is valid and there are more actions in the task as specified in the Task Model. In this case, the prediction will be the next action in the currently predicted task. If the current predicted task is correct and there are no more actions in the task, the next possible tasks are retrieved from the Task Model as illustrated in Figure 5. The Task Identifier decides on the next task based on probability and/or the match between the user's current context and the context associated with the next possible tasks. The predicted action will be the first action in the chosen task. However, if the probabilities are below a threshold value and there is no context match, no prediction can be made. When the current predicted task is invalid, the Task

Identifier searches the Task Model to allocate the action the user actually performed as illustrated in Figure 6. If the action is found, the next action is chosen based on probability and context match. No prediction can be made if the probabilities are below a certain threshold and there is no context match.

The predicted action is sent to Proactivity which in turn sends a prediction feedback to the User Intent Manager indicating whether the User Intent prediction was acceptable to the user or not. Based on this feedback, the User Intent Manager maintains a prediction hit percentage which assists the decision as to when to start a task discovery cycle as described in Section 4.1.

If the previous prediction cycle was successful
If there are more actions in the currently predicted task
Predict the next action in task
Else if the last action in the task has just been performed
Check next tasks in the Task Model
Choose next task based on probability & context match
If there is no context match & probabilities below a threshold value
Inform Proactivity that a prediction can't be made
Else
Predict the first action in the task with highest probability and/or closest context match

Else if previous prediction cycle was not successful
Locate last action performed by the user in the Task Model
If action is found
Check next actions in the Task Model
Choose next action based on probability & context match
If there is no context match & probabilities below a threshold value
Inform Proactivity that a prediction can't be made
Else
Predict action with highest probability and/or closest context match
Else if action not found
Inform Proactivity that a prediction can't be made

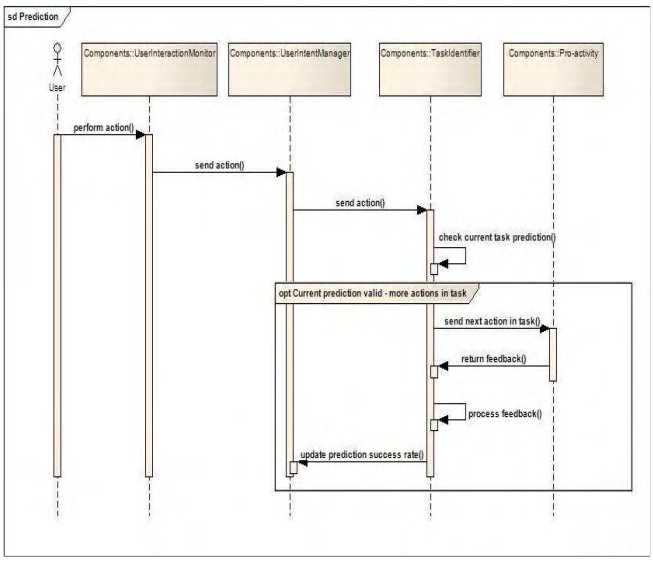

Figure 4 Prediction When Current Predicted Task is Valid with More Actions in Task

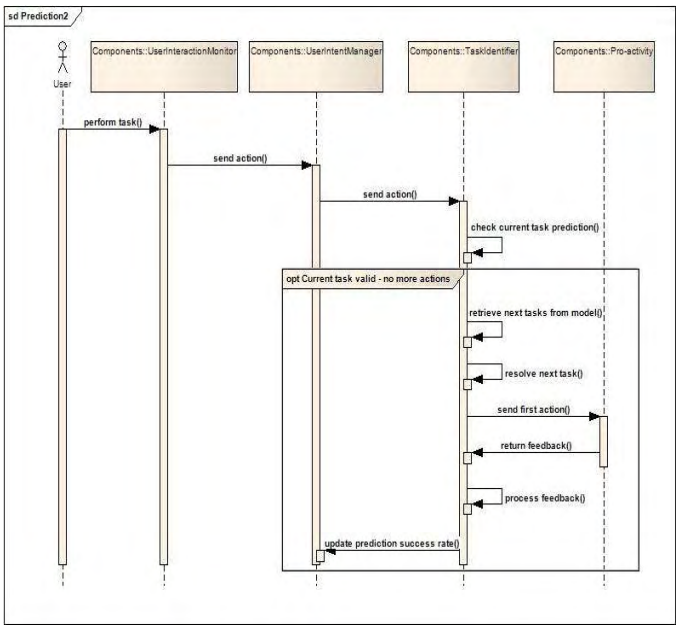

Figure 5 Prediction When Current Predicted Task is Valid with No More Actions in Task

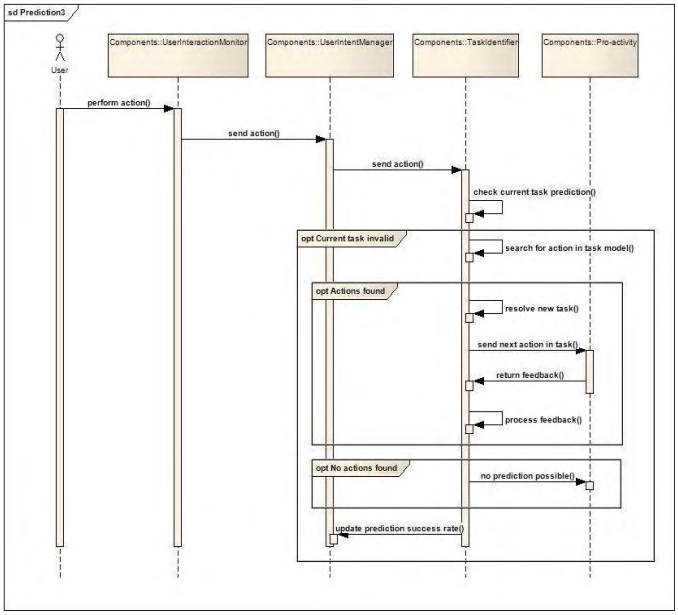

Figure 6 Prediction When Current Predicted Task is Invalid

5 CONCLUSION AND FUTURE WORK

In this paper, the notion of a Personal Smart Space (PSS) is introduced and an overview of the Persist system architecture is given. The User Intent component required to support the proactive behaviour in Persist is described. The architecture of

the User Intent subsystem is explained. A formal definition of the task discovery problem is provided. The use of the task model discovered to predict the user's next intended task/action is described. Associating user context with the tasks discovered and using such context to assist in the prediction of the user's intended task/action is introduced.

A number of open issues still need to be tackled in the User Intent subsystem. This includes 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 required to define the format in which the discovered task model is stored and where this model should be stored. One proposal is to store the discovered task model as a graph in the Prediction part of the User Intent subsystem and periodically update a backup copy in the PSS's database. However, partitioning the model and storing only part of it in User Intent is also a serious option as the discovered task model becomes too large to be accommodated in a mobile device with limited memory capabilities. The use of a priori knowledge of tasks, defined explicitly by the user or based on tasks performed by other users to predict user intent can also be considered. Such tasks could improve the accuracy of User Intent predictions at the early stage of system usage when there is no enough history to learn an accurate task model.

ACKNOWLEDGMENT

This work was supported by the European Union under the FP7 programme (PERSIST project) which the authors gratefully acknowledge. The authors also wish to thank all colleagues in the PERSIST 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 PERSIST consortium. Apart from funding the PERSIST project, the European Commission has no responsibility for the content of this paper.

REFERENCES

[1] S. Yoshihama, P. Chou, and D. Wong, "Managing Behaviour of Intelligent Environments", Proc. PerCom '03, pp 330-337, 2003.

[2] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S.X.Q. Zhang, "The Intelligent Home Testbed", Proc. Anatomy Control Software Workshop, 1999, pp 291-298.

[3] 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, Vol 36(3), 2006, pp. 328 - 340.

[4] K. Gopalratnam, D. J. Cook, "Online Sequential Prediction via Incremental Parsing: The Active LeZi Algorithm", Intelligent Systems, IEEE, Vol. 22(1), 2007, pp.52-58.

[5] A. Kroner, D. Heckmann, and Wolfgang Wahlster, "SPECTER: Building, Exploiting, and Sharing Augmented Memories", 2006.

[6] H. Si, Y. Kawahara, H. Morikawa, T. Aoyama, A stochastic approach for creating context aware services based on context histories in smart Home, Proceeding of 1st international workshop on exploiting context histories is smart environments, Pervasive 2005, Munich ,Germany, May 2005.

[7] H.E. Byun, K. Cheverst, "Utilising Context History to Provide Dynamic Adaptations", Journal of Applied AI, Taylor & Francis. Vol. 18, No. 6, July 2004.

A Hybrid Preference Learning and Context Refinement Architecture

Korbinian Frank[1], **Patrick Robertson**[1], **Sarah McBurney**[2], **Nikos Kalatzis**[3], **Ioanna Roussaki**[3] and **Marco Marengo**[4]

Abstract. Pervasive computing envisions a world where people are surrounded by numerous communication and computing interconnected devices that are invisible and assist users in their everyday tasks in a seamless unobtrusive manner. Most pervasive computing research initiatives aim towards the realization of smart spaces, i.e. fixed spaces that provide pervasive features in a static and geographically limited environment. To bridge these isolated pervasive spaces, the EU project Persist has introduced the concept of self-improving Personal Smart Spaces (PSSs) that follow their owners wherever they go. This paper provides an overview of the context and preference learning facilities that have been designed to support the realization of PSSs and enhance their proactivity and self-improvement features.

1 Introduction

Pervasive computing [22] [21] is a next generation system paradigm that aims to assist users in their everyday tasks in a seamless unobtrusive manner. It assumes that users are surrounded by numerous communication and computing devices of various features, which interoperate and are capable of capturing and processing information regarding users, their behaviour and their environments. This information is used to establish context-awareness features [16] and to enable the provision of personalized context-aware services [12]. In this framework, there have been various research initiatives aiming towards the design and realization of smart spaces [23] in homes, offices, universities, schools, hospitals, hotels, museums, and other private or public places, where various automation facilities support the users. Nevertheless, these are fixed spaces that provide pervasive features in a static and geographically limited environment. They are alike independent "islands" of pervasiveness in a sea of legacy service provisioning systems. When the users exit these "islands" no pervasive computing features are offered. To bridge this gap, the notion of self-improving Personal Smart Spaces has been introduced by the Persist project.

Persist is a European research project (`http://www.ict-persist.eu/`) funded under the Seventh Framework programme that aims to couple the facilities offered by next generation mobile communications with the features provided by the static smart spaces to support a more ubiquitous and personalised smart space that is able to follow the user wherever he/she goes. A Personal Smart Space (PSS) will provide to its owner multimodal intelligent interfaces, via which he/she will be able to access and configure the various services and resources that are available locally and remotely, even when limited or even no network connectivity is available. PSSs will be able to discover other PSSs and interact with them in order to create a richer and more flexible environment for their owners. Each PSS consists of multiple devices, both mobile and fixed, owned by a single user. As the owner of the PSS moves to different locations and places, his/her PSS interact with other mobile or fixed PSSs located in the owner's surrounding environment, aiming for a unique support for pervasive service provisioning. PSSs constantly monitor their owner's behaviour & environment and they exploit learning techniques to further optimise the pervasive experience perceived by their owner's. In a nutshell, a Personal Smart Space can be defined as a set of services within a dynamic space of connectable devices, where the set of services are owned, controlled, or administered by a single user. It facilitates interactions with other PSSs, it is self-improving and is capable of pro-active behaviour. Thus, a PSS is user centric and controlled by a single user, it is mobile (at least from user perspective), it allows interactions with other PSSs and is capable of self-improvement and proactivity.

In order to establish the proactivity and the self-improvement features of PSSs, quite sophisticated learning facilities are required. These facilities need to support both the learning of the user preferences, in order to enable the provision of personalized services, as well as the inference of future or currently unavailable context information. This paper elaborates on the mechanisms that have been designed in order to support context and preference learning and automated extraction.

The rest of this paper is structured as follows. In Section 2, a library-based hybrid architecture for preference learning and context refinement is presented, which is designed so as to demonstrate properties such as: transparency, modularity and pluggability. Section 3 elaborates on the context refinement algorithms. In this respect, the following are described: Bayesian Filters that refine the location accuracy, a clustering technique used to discover recurring locations, a history-based context prediction mechanism, Bayesian high-level context inference and a proximity estimation method that is based on a diffusion model. Subsequently, in Section 4, three different preference learning algorithms are described: the if-then-else rule learning, the online incremental preference learning and the Bayesian learning of user behaviour. Finally, in Section 5 conclusions are drawn and an overview of further research challenges regarding learning in personal smart spaces is provided.

[1] German Aerospace Center (DLR), Germany, email: {korbinian.frank | patrick.robertson}@dlr.de
[2] Heriot-Watt University Edinburgh, UK, email: S.M.McBurney@hw.ac.uk
[3] National Technical University of Athens, Greece, email: {nikosk | nanario}@telecom.ntua.gr
[4] Telecom Italia s.p.a., Italy, email: marco.marengo@telecomitalia.it

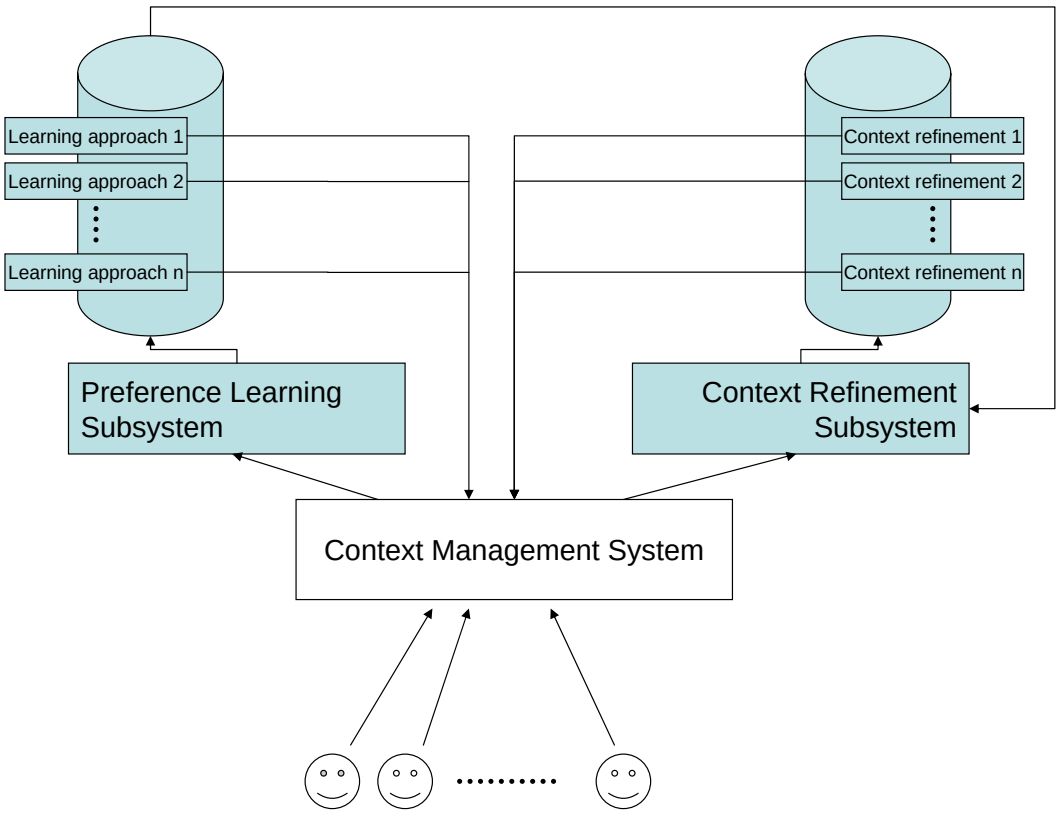

Figure 1. Simplified Architecture model for Preference Learning and Context Refinement libraries

2 A Library based Architecture Model

This section will present our hybrid architecture for preference learning and context refinement. It is part of the software architecture of the EU project Persist. It only sketches the relevant part of the overall architecture that can be found in [6].

The architecture section presented in this paper is shown in Figure 1. It represents a crucial part for realising self-improvement in this system. In particular it interacts with the Context Management subsystem and the Preference Management system, the access points to external usage. while itself is only called from within the Persist architecture. It is designed by the following guidelines:

- *Transparency*: The system user does not have to have any knowledge about the existence of this system and should not be aware either when it runs in the background. Response times have to be fast and computational extensive processes should run at times with low processor occupancy. All queries have to be issued to a single access point.
- *Modularity*: Deployment on mobile devices has to deal with limited resources. Therefore some processes will have to be performed in the back end of the Persist system, while other modules are deployed on all mobile devices. Therefore our architecture has to allow for a splitting of functionality and has to be able to run even when some parts are missing locally or are not reachable due to a lack of network connectivity.
- *Pluggability*: There are many different approaches for machine learning and refinement of context information. This architecture shall not reduce the performance capabilities of the Persist system by limiting to a certain number of algorithms for self improvement. Even during runtime it has to be able to include new algorithms that can immediately be used further on. This is allowed by a common interface for all methods within one library.

As you can see in Figure 1, the presented part consists of two interrelated library systems. On the left, the *Preference Learning subsystem* is strongly related with the Preference Management system as it learns and updates preference rules for this system. It is triggered by the Persist Eventing system upon user actions that are stored in the context database. These actions are retrieved by the Preference Learning subsystem and passed on to the methods in the learning library which use it either directly or after a certain time or amount of gathered actions. The resulting preference rules are stored again in the context database, but may also serve as input to the context refinement subsystem that are specialised on evaluating them.

Hence the right branch of the picture is dedicated to evaluation algorithms. Context refinement can be any method that accesses available context information from the Context Management system and refines/enriches it. Therefore a number of different algorithms are inside the context refinement library. These can work either on demand or run continuously, subscribing themselves for changes of their input context information. While on-demand algorithms are triggered by the Context Management system, if requested information is not available, continuous refinement can be caused either also by context consumers that have to be notified upon any change or by the nature of the approach.Continuous refinement would be in theory desirable for any client, but practice shows that it is much more resource consuming and often needless. So a suitable trade off has to be found.

3 Context Refinement algorithms

Context information gathered by sensors is stored and managed in the Context Database, as shown in the last section 2. In many cases however this information is not easily processable for context consumers, as it is too fine grained, not reliable enough, not meaningful or also just not the information that is searched for. That is why a context aware architecture must provide a number of approaches to refine the raw context information and provide the consumers with the right, reliable and precise information.

3.1 Bayesian Filters to refine Location Accuracy

Location estimation of a moving entity such as a person can be viewed as a static or dynamic estimation process, depending on how often a location estimate is needed, or can be efficiently generated. It can be shown that the dynamic Bayesian estimator is the most accurate form of estimation for this problem and very often algorithms from the family of particle filters are applied, [9]. The reason for the superiority of dynamic filtering over repeated one-shot estimation is that the former makes use of the temporal correlations of the location in the form of a process or "motion" model. When location updates are only needed rarely then the cost of continual estimation - both in terms of battery power and any other resources such as communications bandwidth - must be weighed against the improved accuracy. Also, dynamic estimation is far more suited to estimate the full pose of a person, i.e. also the direction the person is facing.

In sequential Bayesian filtering, sensors provide some information related to location or motion. Such sensors can encompass GPS receivers, RFID readers, WLAN sensors, UWB systems, barometric altimeters, magnetometers, inertial sensors and many more. It is important to recognize that these sensors do not necessarily need to give location information directly; a magnetic compass gives information about the orientation of a person which still allows positioning itself to be improved in a dynamic scenario [25]. Furthermore, map information can be used to improve positioning, especially when using inertial sensors. Hence "refinement of location accuracy" is an informal but frequently used term for the process of combining any sensor information with the goal of accurate location information.

Dynamic localisation is especially important in the indoor environment since many sensors that are usually very accurate no longer perform very well, such as GPS receivers or mobile radio positioning. dedicated infrastructure such as UWB systems may be available but suffer from coverage problems. In these cases a motion model in a dynamic estimator that takes maps into account can greatly improve performance.

Finally, an important advantage of any Bayesian filter is that it inherently provides uncertainty information about the location and pose, which can be passed to other processing layers, such as probabilistic context inference.

3.2 Clustering to discover recurring locations

Current terminal devices can readily access location related information whereas other sources of contextual information are harder to gather and process, and many techniques may be applied to enhance the accuracy of such data. Nevertheless, coordinates identify a single point in space, while people are most accustomed to reason in terms of "places", regions of space. Location tagging is a method for inferring *"places"* from a set of positions. "Places" are regions of space that carry some meaning to the user and to which the user can potentially attach some (meaningful) semantics. Examples of places include home, work, pub and airport.

Figure 2. From *positions* to *places*: how a set of coordinates (left) is transformed into places (right)

The location tagging component is part of the reasoning framework and offers both places detection functionalities and a user interface for managing the user location model. Once discovered, places are defined by a region of space, characterized by a non null area, a description and a place-type which is chosen from a simple place-types taxonomy [10].

The learning process involves the following steps:

- *clustering*: user position history is partitioned in several clusters of data, which represent the most recurring locations. Our implementation uses the Shared Nearest Neighbour (SNN) [15] algorithm to discover clusters with different densities.
- *place definition*: once discovered, clusters are just sets of points, making it difficult to store and to use them. A convex hull algorithm [3] is therefore applied to each cluster in order to discover the smallest polygon that includes all the cluster's points. Only the vertices of each cluster are stored.
- *user validation*: before being useful, places need to be validated by the user. Users are able to manipulate their places and eventually add new ones or delete unwanted ones. Assigning place-types (e.g. *"office"*, *"restaurant"*, ...) to a user's places enables exciting scenarios, as other reasoning components could make use of such information to offer place-aware services or advertisements.

The clustering step is time-consuming and should therefore be carefully scheduled. It is also recommended to follow an incremental approach, in order to reduce the amount of processed data.

3.3 History based context prediction

The efficient collection, maintenance, and processing of *History of Context (HoC)* [13] is critical for Personal Smart Spaces, as it allows for inference of future context and current context information that is no longer available. The HoC management framework in Persist, also caters for elements of the user behaviour, thus providing input to support the user preferences' learning process, as well as the inference of user intent regarding the usage of pervasive services. The designed HoC management facilities are also able to support the extraction of periodic patterns regarding context data combinations and use these patterns in order to deliver successful context predictions.

The proposed approach regarding the HoC lifecycle consist of 4 generic phases which are schematically depicted in figure 3. Initially, context sources feed the measured context data in the main context DB synchronously (Step 0) or asynchronously (Step 1). Subsequently, this information is cached in the HoC DB maintained

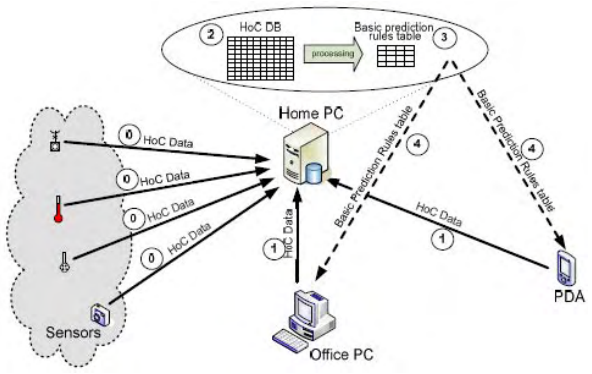

Figure 3. History of Context lifecycle

by a resource-rich system (e.g. a user's home PC) where it is processed and scored(Step 2). Based on this input and enforcing time-dependent attenuation to all past context values, context prediction rules (called Basic Prediction Rules) are generated (Step 3) and disseminated to PSS systems supporting HoC based context prediction or estimation (Step 4).

In more details the PSS user or administrator may identify the context data types that need to be monitored. The values of HoC data recorded can be quantifiable, such as the GPS coordinates expressing location, or can be represented by abstract tags, as is the case for activity (e.g. busy, free, sleeping), device used (PDA, laptop, desktop), semantic location (home, work, restaurant), etc. These data along with the respective timestamps are initially cached at the devices connected with the context sources and when possible forwarded to a resource rich system that maintains the HoC database. The database is structured in 48 half hour time frames. Context data combinations are assigned to time frames depending on the start and end time these were observed. Subsequently, the entries assigned to each time frame are "scored". Thus, instead of maintaining all occurrences of each context data combination, as well as their start/end time and date, each such combination is assigned to a single entry in the corresponding time frame in the HoC database, along with a single score that is decreased as time passes by and increased every time the exact same context situation is observed.An example segment of the main HoC DB table is presented in figure 4.

TF Index	entityID-location-device-activity	Score
0	…person#471-Home-NONE-Sleeping	77.74255339
	…person#471-Home-TV-WatchingTV	13.32932862
	…person#471-Pub-PDA-Drinking	4.866635658
	…person#471-Restaurant-PDA-Eating	2.491089311
1	…person#471-Home-NONE-Sleeping	79.46278259
	…person#471-Home-TV-WatchingTV	15.57818485
	…person#471-Pub-PDA-Drinking	2.002311926

Figure 4. Example of part of the main table of the HoC database

This approach aims to reduce the storage resources required and achieve significant context summarization. The score values are calculated on a daily basis. Depending on the score values and the selected rule generation model, context prediction rules are generated. As the size of the prediction rules is minimal, it is possible to forward them to all the remote devices enabled for context prediction or inference, irrespective of the storage and processing resources they have available.

3.4 Bayesian High-level Context Inference

The last sections 3.1 and 3.2 showed how to add meaning and precision to context information by refining sensor readings, respectively deducing missing context information from past patterns in section 3.3. For many applications however context information is necessary that cannot come from sensors – as there *is* no appropriate sensor: for so called high-level context information. Most prominent among those are *situation*, *activity*, *mood*, *manoeuvre*, *comfort*, *danger*, *stress* and so on.

If an application or a preference depends on such high-level information ("IF `user is going home`..." or "IF `user is not stressed`...") this information has to be inferred from available context source information by appropriate rules.

A very promising approach for this inference is the use of *Bayesian Belief Networks (BN)* as the rule representation. BNs consist of random variables (represented as nodes) with defined states which are interconnected by directed edges representing causal dependencies and contain a conditional probability distribution (CPD) [19, 11]. The random variables can represent context attributes, their states model the possible values of the context attribute, as can be seen in Figure 5. If sensor readings are available, the respective at-

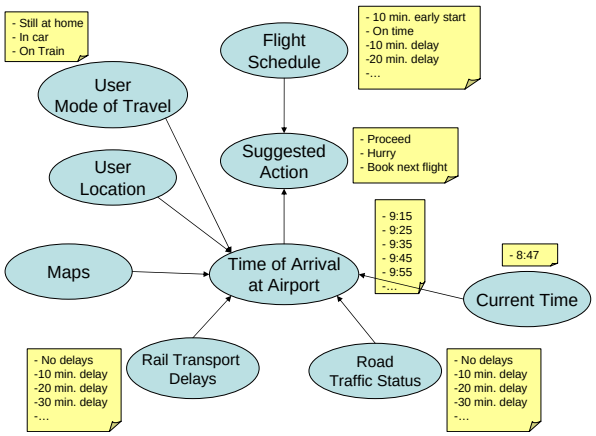

Figure 5. An example Bayesian Network

tributes are assigned *evidence*, which influences the probabilities of related attributes – calculating the most probable state of a target context attribute given all available evidence can be used as context inference. This approach combines several advantages [2]:

- A Bayesian network readily handles situations where some data entries are missing or uncertain.
- As it has both a causal and probabilistic semantic, it is an ideal representation for combining prior knowledge and data.
- Bayesian statistical methods in conjunction with Bayesian networks avoid the overfitting of data and do not need to separate data into training and testing sets. They are also able to incorporate smoothly the addition of new data as it becomes available.

One drawback of this kind of inference is its time complexity. Cooper [4] identified the problem as NP hard in the number of random variables, which holds even for singly connected BNs (see [26]). With a large number of interrelated context attributes, context inference could therefore become unfeasible in practice on mobile, resource limited devices. To overcome or reduce this problem, either the number of nodes or the number of states per node or the number

of edges have to be limited. Our approach [7] is using *Bayeslets* to reduce computation time by reducing all three parameters.

Furthermore it offers us the chance to personalise inference rules, which is of particular importance for high-level context, as it depends very much on the user's individual perception. Inference rules based on Bayeslets can even adapt during run-time to changed behaviour of its owner. If the Context Management System triggers inference on a particular context attribute, it chooses the Bayeslet of the target user that has the desired context attribute as output node, connects it to the related Bayeslets to build a complete, relevant inference rule, evaluates it and returns the inferred state of the requested context attribute.

3.5 Proximity Estimation with a Diffusion Model

Consider the following short use-case: A person is visiting a company for a meeting and would like to print a document. She is not very familiar with the layout of the premises and there are certain restrictions as to which printers are allowed to accept and process requests from visitors. Assuming that the visitor is associated with the local network of the premises a context dependent search for suitable printer services can take place. This search can encompass not only administrative constraints, and those based on the service usage (e.g. for a colour printer), but also on location and proximity metrics. In particular, our visitor should usually be guided to the nearest suitable printer, where nearest refers to reachability in pedestrian terms.

Simply applying a straightforward Euclidean metric in 3D space will usually fail, since non-linearities, such as walls, in the true practical proximity are not considered. A true proximity metric should be proportional to something like the time taken to reach the destination or the distance travelled.

In [1] a mobility model pedestrians was introduced that applies a gas diffusion algorithm to estimate a path between two points in a given floor plan layout. The algorithm works by emitting a virtual "gas" source from the destination that propagates through the open portions of a building layout and becomes absorbed by walls. Any point in the building experiences a certain concentration of the "gas" that is proportional to the distance to the emitter (destination, e.g. a particular printer). So, physical facilities like printers can be easily rated by their diffusion matrix by comparing their relative "gas" densities at the starting point (e.g. where the user is located). Computation of the gas source is straightforward ([14]) and the results may be cached in a database.

Once a user has selected a suitable destination, using the diffusion model and other criteria such as suitability, the model can also be used in a very simple way to generate routing information to guide the user to the destination. All that needs to be done is to follow the steepest gradient of "gas" concentration to the destination.

In order to impose restrictions in motion for certain people (e.g. visitors may not be allowed access to certain areas) then these restrictions can be represented in their own layout. People with disabilities may also impose restrictions on certain features of a building, such as stairs.

4 Preference learning algorithms

Creating and maintaining a set of user preferences is a continuous and important task as the quality of the personalization provided depends on the quality and completeness of the user's preference set. When a new user enters a pervasive environment or an existing user starts to use a new service, network or device existing preferences may need to be refined or new preferences added. Performing this task in an entirely manual fashion is undesirable for two reasons. Firstly, the possible number of personalizable attributes (e.g. service customization parameters, service selection criteria, etc.) is potentially very large with each possibly requiring a preference. Manually creating and maintaining such a large set requires time and effort, detracting from the benefits that personalization aims to provide. Secondly, some user behaviours may be so implicit that users may not identify them as explicit preferences. For example, a user may not be aware that he/she always selects a service within a certain price range at a certain time of day. Even if the user was aware of such a behaviour it may be difficult for them to represent it in the appropriate format. Therefore it is essential that mechanisms, such as monitoring and learning are in place to aid the user with preference creation and maintenance, providing them with a more accurate and complete preference set and hence better overall personalization.

4.1 IF-THEN-ELSE Rule Learning

User preferences can be represented in many ways however, it is beneficial to provide a human-understandable format to allow the user some control over their preference set and hence personalisation. One adopted approach is to express user preferences as IF-THEN-ELSE rules to allow for human-readability. One way to achieve this is to first learn a decision tree, then translate the tree into an equivalent set of rules. The IF-THEN-ELSE rule learning algorithm [17] implements this strategy. It is an adaptation of Quinlan's C4.5 batch learning algorithm [20] which aims to split outputs into distinctive groups using a minimal number of attributes from the input dataset. As with C4.5 Gain Ratio is used to combat problems arising from attributes with multiple values.

Input to the algorithm is a list of `userAction` objects each containing some action the user has performed (e.g. `video = paused`) and a context snapshot of the user's context situation when they performed the action (e.g. `location = home, call = incoming`). Each userAction also contains further meta-data such as the service type and service instance from which the action originated. The list of userActions is stored in the Context Management System in the User History Database. Learned output is a set of decision trees (one for each learned preference) with context attributes as condition nodes and actions as leaf nodes. Each tree is then mapped to IF-THEN-ELSE rules for human-readability.

The dynamic nature of pervasive environments often challenges the batch approach to preference learning. For example, if the user changes his behaviour this cannot be captured until the next batch learning process. Even then, new behaviour may be over-written by more established (but perhaps distant) behaviours. Several projects [24] implement 'quick response' mechanisms that immediately accommodate new behaviours into the user's preference set however this approach can suffer from catastrophic forgetting which is undesirable if the behaviour change was only temporary.

To overcome these challenges a novel dual store approach is adopted for the User History Database. This acts as a *short-term store (STS)* and *long-term store (LTS)*. Figure 6 below illustrates the content of each partition.

The STS only holds userActions which have happened since the most recent batch learning process at time t_i (i.e. this is where new userActions are initially stored). The data held in the STS is the input to each batch learning process. After the next batch learning process occurs at time t_j the STS will be retrieved, learned upon, copied

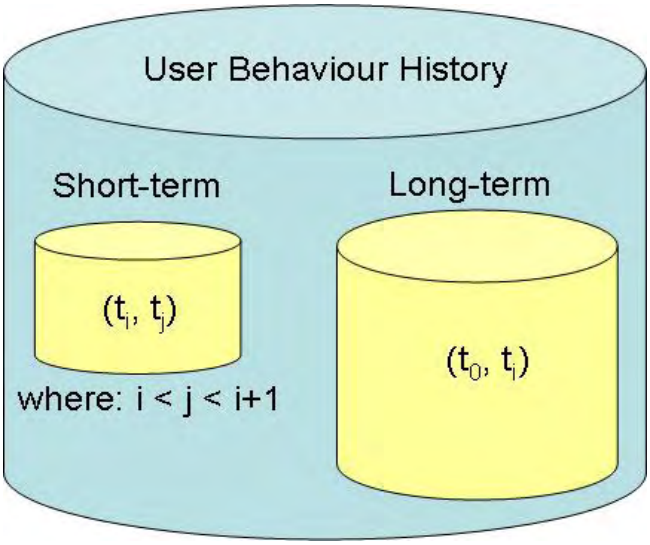

Figure 6. Dual User History

to the LTS and cleared. The LTS stores all user actions which have happened since the user started to ever use the system at time t_0 until the most recent batch learning process at time t_i. It acts as a backup containing the entire back catalogue of `userActions`. The LTS is used to resolve any conflicts that may arise when merging new learned information with existing preferences. The dual store approach allows recent changes in user behaviour to be identified quickly while also providing support where behaviour changes are temporary or where conflicts occur between new learned information and existing preferences.

4.2 Online, Incremental Preference Learning

As a compliment to the batch IF-THEN-ELSE learning algorithm, Persist will also employ on-line, incremental preference learning. This will be implemented as a hetero-associative neural network that will associate current context to user actions. The topology of the neural network allows for internal network knowledge to be translated into IF-THEN-ELSE rules for user understanding. This approach will be described in detail in a forth-coming paper.

4.3 Bayesian Learning of Behaviour and Inference Rules

As described in section 3.4 Bayesian Networks (BN) can be used to represent the probabilistic relationships between random variables that represent various aspects of context information and user actions, from sensor readings to the high-level context attributes such as a person's current activity. Such a network structure is defined completely by the set of random variables (nodes) and directed arcs between random variables that can often be interpreted as causality (in the direction of the arc) [19]. The conditional probability distributions of each node - conditioned on each possible instantiation of all the parent nodes - defines the network quantitatively (parameters of the network). As described earlier, given the network structure and parameters inference is the process of computing the probability of a set of hidden random variables given observations of other random variables. This section will briefly address the approaches used to obtain both network structure and parameters using a combination of expert knowledge and previous observations of the random variables by *learning*.

Applied to learning of behaviour rules, some RVs (preference outcome nodes) can either be interpreted as active behavioural choices by the user, or the BN can be extended to a utility network with decision and utility nodes, where the decision nodes are the outcomes. Applying such learnt rules during the use of the system is thus the process of evaluating known context information to infer the user's most probable choices or actions based on learning the BN (which itself can be interpreted as a probabilistic behavioural rule).

4.3.1 Determination of the scope of the network

Context information pertaining to a user of Smart Electronic Spaces may in general encompass a very large range of human activities, sensor readings, information such as calendars, as well as such data relating to other people; all represented as random variables (RVs) in a very large BN. Obviously it is impossible to include all sets of such RVs in a representation used in inference, as the resulting BN would be too large. Take, for instance, the case where we are attempting to infer the current activity of a person in an office environment. Her activity will be part of a BN that includes not just her location and pose but also information such as the devices she is using, her calendar schedule, her heart rate and other bio-sensor readings, her past activity, the time of day, proximity to other people, the status of projects she is working on, as well as much more; additionally, such a BN will encompass similar aspects of other people's context, such as that of her colleagues. For practical relevance it will be necessary to impose boundaries on which RVs will be used to represent the information "around" her activity. For this purpose we have previously introduced the concept of "Bayeslets" that are small BNs that may be linked to form larger BNs [7]. It is fair to assume that human expert knowledge will be used to determine the RVs that make up such a Bayeslet, and which smaller Bayeslets can be assembled to represent a problem domain such as activity.

4.3.2 Learning static BNs

We have thus far restricted our approach to encompass only static BNs, rather than dynamic BNs [18], and to RVs with discrete variables. To incorporate at least some temporal aspects, we do allow RVs that encode previous outcomes of RVs, such as a user's activity or location a certain time ago or in the future. Assuming that a human expert has determined the scope of RVs in a learning process we can resort to established BN learning techniques for:

1. Learning with complete data.
2. Learning with missing data.
3. Learning with missing data and also hidden variables.

Learning with complete data sets of all involved RVs is easiest and follows the approach described in [5]. Usually some form of greedy hill climbing is used to rate candidate networks and we take the best scoring NW ignoring network structure priors. Naturally, we will impose knowledge of:

- Causality between RVs (such as sensors readings being caused by a physical process); i.e. the presence and direction of arcs.
- Arcs that we know or assume are missing between RVs, thus imposing more independence.
- Complete sub-units of the network that are assumed to be known.

For missing data we will employ algorithms from the class of (structural) expectation maximization (EM) introduced in [8].

5 Conclusion and Outlook

This paper reflects the work being undertaken in task 4.4 of the EU project Persist that is investigating learning and reasoning algorithms for personal, self-improving smart spaces. Learning preferences and reasoning rules is a core part for the self-improvement targeted in Persist, inference and in general all context refinement procedures are the methods necessary to give expressivity to preferences and to evaluate them.

We decided to adopt a library based approach with a single access point and common interchange formats, that gives access to a collection of different algorithms for learning and context refinement. This library is not limited however to the presented methods, but open and extendible to new approaches allowing for the same interface.

This paper presents a number of context refinement approaches that apply on different levels. Fusing raw sensor data for more precise information in section 3.1, then the enrichment of this precise absolute location with semantics like in section 3.2. Such meaningful information is basis for the approaches for context prediction in section 3.3 and high-level context inference in section 3.4. A particular high-level context information, proximity, can be evaluated with the approach shown in section 3.5.

We have presented in this work furthermore three different algorithms for learning, each offering some variants. Differences between them are the execution procedure (batch vs. incremental learning), weighting of long past and recent input information and also the purpose. The approaches in sections 4.1 and 4.2 are identifying behavioural patterns for preferences whose conditions are permanently monitored further on, while the probabilistic approach in section 4.3 can provide Bayesian context inference rules (suitable for the approach in section 3.4) as well. These approaches are implemented in the project Persist that will publish its source code at an Open Source portal as a basis and guideline for fellow researchers. Still there are a number of questions to be answered.

- A particular challenging one will be the orchestration of these approaches. A component that delegates learning or context refinement tasks to the most appropriate approaches is not considered so far.
- Neither has there been undertaken detailed work about conflict resolution mechanisms yet. In case of contradicting results from different methods this will be necessary. This functionality could be based on confidence of the methods' outcomes that is evaluated by a probabilistic component, e.g. in a static BN.
- Resource and computation time issues of the overall system have not been dealt with so far.
- Privacy issues are not treated in this paper, but are part of the Persist architecture. The outcomes of this research is published in a separate paper.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 215098 of the *Persist* (PERsonal Self-Improving SmarT spaces) Collaborative Project

REFERENCES

[1] M. Angermann, J. Kammann, and B. Lami, 'A new mobilty model based on maps', in *The 58th IEEE Semiannual Vehicular Technology Conference (VTC Fall 2003)*, (2003). Date=2003-10-06 - 2003-10-09;.

[2] M. Angermann, P. Robertson, and T. Strang, 'Issues and requirements for bayesian approaches in context aware systems', in *LoCA 2005*, ed., C. Strang, T.; Linnhoff-Popien, (05 2005). event_dates=2005-05-12;.

[3] H. Huhdanpaa C. Barber, D. Dobkin, 'The quickhull algorithm for convex hulls', *ACM Transaction on Mathematical Software*, **22**(4), 469–483, (1996).

[4] G. F. Cooper, 'Probabilistic inference using belief networks is NP-hard', Technical Report KSL-87-27, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, CA, (May 1990).

[5] G. F. Cooper and E. Herskovits, 'A bayesian method for the induction of probabilistic networks from data', *Machine Learning*, **09**(4), 309–347, (October 1992).

[6] I. Roussaki et al., 'Initial architecture design', Technical report, PERSIST, FP7-ICT-2007-1, (November 2008).

[7] K. Frank, M. Röckl, and P. Robertson, 'The bayeslet concept for modular context inference', in *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM08)*, eds., J.L. Mauri, N. Cardona, K. Chen, M. Popescu, and A. Doci, pp. 96 – 101. IEEE Computer Society Conference Publishing Services (CPS), (05 2008). event_dates=[2008-09-29 - 2008-10-04];.

[8] Nir Friedman, 'The bayesian structural em algorithm', in *In UAI*, pp. 129–138. Morgan Kaufmann, (1998).

[9] F. Gustafsson, F. Gunnarsson, N. Bergman, and Forssell U. et al., 'Particle filters for positioning, navigation and tracking', *IEEE Transactions on Signal Processing*, (2002). vol. 50, No. 2, 2002.

[10] H. Tschofenig H. Schulzrinne. Rfc 4589: Location types registry. IETF.

[11] D. Heckerman, 'A tutorial on learning with bayesian networks', Technical report, Learning in Graphical Models, (1995).

[12] K. Henricksen and J. Indulska. Personalising contextaware applications, 2005. K. Henricksen and J. Indulska, Personalising ContextAware Applications, in OTM Workshop on ContextAware Mobile Systems, vol. 3762, Lecture Notes in Computer Science: Springer-Verlag, 2005, pp. 122-131.

[13] Nikos Kalatzis, Ioanna Roussaki, Nicolas Liampotis, Maria Strimpakou, and Carsten Pils, 'User-centric inference based on history of context data in pervasive environments', in *SIPE '08: Proceedings of the 3rd international workshop on Services integration in pervasive environments*, pp. 25–30, New York, NY, USA, (2008). ACM.

[14] Mohammed Khider, Susanna Kaiser, Patrick Robertson, and Michael Angermann, 'A novel movement model for pedestrians suitable for personal navigation', in *ION NTM 2008*, pp. 819 – 827. The Institute of Navigation, (01 2008). Date=2008-01-28 - 2008-01-30;.

[15] V. Kumar L. Ertöz, M. Steinbach, 'Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data', *SIAM International Conference on Data Mining (SDM '03)*, (2003).

[16] Seng Loke, *Context-Aware Pervasive Systems: Architectures for a New Breed of Applications*, Auerbach Publications, 2006.

[17] S. McBurney, E. Papadopoulou, N. Taylor, and H. Williams, 'Adapting pervasive environments through machine learning and dynamic personalization', in *Proc. of the 2008 Conference on Intelligent Pervasive Computing*, pp. 395–402, (2008).

[18] Kevin P. Murphy, *Dynamic bayesian networks : representation, inference and learning*, Ph.D. dissertation, 2002.

[19] Judea Pearl, *Causality: Models, Reasoning, and Inference*, Cambridge University Press, 2000.

[20] John Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kauffman, 1993.

[21] D. Saha and A. Mukherjee, 'Pervasive computing: a paradigm for the 21st century', *Computer*, **36**(3), 25–31, (2003).

[22] M. Satyanarayanan, 'Pervasive computing: Vision and challenges', *IEEE Personal Communications*, **8**, 10–17, (2001).

[23] Ramesh Singh, Preeti Bhargava, and Samta Kain, 'State of the art smart spaces: application models and software infrastructure', *Ubiquity*, **7**(37), 2–9, (2006).

[24] SPICE. Service platform for innovative communication environment. http://www.ist-spice.org/, 2008.

[25] K. Wendlandt, M. Khider, M. Angermann, and P. Robertson, 'Continuous location and direction estimation with multiple sensors using particle filtering', in *MFI 2006*, ed., IEEE. IEEE Verlag, (09 2006). Date=2006-09-04 - 2006-09-06;.

[26] D. Wu and C. Butz, 'On the complexity of probabilistic inference in singly connected bayesian networks', in *RSFDGrC(1)*, pp. 581–590, (2005).

Giving the User Explicit Control over Implicit Personalisation

Sarah McBurney[1], **Nick Taylor**[1], **Howard Williams**[1] **Eliza Papadopoulou**[1]

Abstract. Personalisation has an important role to play in a pervasive environment, supporting resource management tasks and tailoring the system to behave in ways that suit the user. However, this depends on creating and maintaining an adequate set of information on the user's preferences. Experience has shown that one cannot simply rely on the user to input preferences, and it is essential to employ some form of machine learning to support this; on the other hand the user needs to be able to control this. Two major approaches used for this purpose are rule-based strategies and artificial neural networks (ANNs). Within the Persist pervasive system these two different approaches are being combined to give maximum benefit. However, in order to enable the user to maintain control over the resulting set of user preferences, it is essential that he/she can see the current state of this preference set and change it whenever this is required. This paper describes how this will be achieved.

1 INTRODUCTION

Since the initial ideas of ubiquitous and pervasive computing were put forward [1] over 16 years ago, much effort has been invested in achieving them. As the costs of devices continue to fall dramatically, the number of different networks open to the user rises and the number of available services soars, it is increasingly apparent that the goal of pervasive environments with ubiquitous access to services, networks and devices comes at a price. The burden on the user to manage resources becomes great unless the system provides adequate support. This is done through some form of personalisation. By this we mean that the system retains knowledge about the user's preferences in an appropriate form and applies this in the decision making process so that the system behaves differently for different users according to their needs.

Early personalisation systems [2,3,4] relied on the user to provide and manage their preference set manually. Although such *explicit* personalization may mitigate resource management to some degree, the burden of manually managing an entire preference set means that user involvement is merely shifted to preference management responsibilities. For example, each time a new service, network or device is encountered in a pervasive environment the user's existing preference set may need to be refined or new preferences added.

For this reason, many pervasive and ubiquitous projects [5,6,7] include machine learning techniques coupled with user behaviour monitoring systems to provide *implicit* personalisation where preferences are created and managed on behalf of the user. These automatic learning techniques fall into two broad classes – those that are based on data structures that are meaningful to the user (generally some form of rule-based representation) and those that are not (generally some form of network representation). The former have the advantage that the user can be engaged in the process of building up the set of preferences, manually changing preference rules that have been wrongly inferred or adding or deleting rules as needed – thereby using both implicit and explicit approaches to complement one another, reducing the need for user intervention but allowing the user full control of their preferences when required. The second class of techniques (including neural nets and Bayesian approaches) can be more efficient for some decisions but the user simply has to accept the result and cannot view or change the data structure created.

However, for some decisions rule based approaches are more useful while for others network based ones are preferable. The Daidalos project [8], which developed a pervasive system aimed at the mobile user, and the Persist project [9], which is developing a pervasive system based on Personal Smart Spaces (PSSs) [10], both employ a combination of automatic learning techniques combined with user involvement. One aim was to achieve the optimal situation in which the learning algorithm can update the preference data structure whenever a change is learnt and the user can inspect the preferences and change them whenever he/she needs to. This is straightforward in the case of rule-based approaches, but it can also be achieved in the case of a neural net using an appropriate form and algorithms to convert the neural net to a set of preference rules and vice versa. The goal of this paper is to present this process.

The next section describes the rule-based preference format used in Daidalos and explains how the Persist project will utilise these rules as well as providing preferences stored as neural networks. Section 3 outlines the problem of translating neural network knowledge into human readable form and sections 4 and 5 propose a possible solution that will be implemented in the Persist project. Section 6 provides a conclusion.

2 PREFERENCE MANAGEMENT

In the rule based approach that was used in Daidalos, and will be used in Persist, user preferences are represented by an IF-THEN-ELSE rule format and are therefore human-readable. Most preferences are context-dependent although context-independent preferences can also be represented. The condition part of an IF-THEN-ELSE consists of a number of primitive conditions, generally tests on context attributes, combined by logical operators (AND, OR). The THEN and ELSE parts consist of outcomes, in the form of actions, or a nested IF-THEN-ELSE.

[1] Dept. of Mathematics and Computer Sciences, Heriot-Watt University, EH14 4AS, UK. Email: {**ceesmm1**, nkt, mhw, ceeep1}**@macs.hw.ac.uk**.

The outcome is followed when the condition is met (e.g. if the outcome is an action it may indicate that a service is started or a video stream is paused). A full BNF description of the preference format is presented in [11].

In order to assist the user in getting started, stereotypes may be used to create an initial set of preferences. By selecting appropriate stereotypes, a set of preferences can be generated, which provide an approximation to what the user wants. Thereafter automatic learning techniques are used to update this set. The user can also view the rules at any time and change them as needed. The learning technique used was an adapted version of Quinlan's tree building algorithm [12], the output of which can be easily translated to the human-readable preference rule format.

The Persist pervasive system will extend the approach used in Daidalos, using both a rule-based approach and a neural net (ANN). This will provide maximum flexibility in that either technique may be used for handling any subset of preferences. For example, the preferences for selecting a service, deciding on which device or network to use, or personalising a third party service, are all different and for each different techniques may be more suitable.

However, in order to maintain the ability for the user to interact with the preferences a more sophisticated ANN learning algorithm (to be fully described in a forth-coming paper) will be used for preference learning.

3 PROBLEM DESCRIPTION

Interpreting the Persist ANN into the human-readable preference format mentioned above is the challenge of rule extraction from neural networks. Although this area has received much research it seems extraction techniques have rarely been used in pervasive environments to present learned preferences to users. This may be due to the different goals of each field. The general aim of rule extraction research is to better understand neural network behaviour by extracting a rule based explanation of network functionality that could be used to create better classification systems. Interpretation of network knowledge comes as an aside. Our aims in the pervasive domain are more user-centric and focused on performing two-way interpretation (i.e. from network weights to rules and back again). Several challenges arise from this user-centric, pervasive problem domain.

One significant problem is ensuring that the extracted preference set is devoid of duplicates and large, over-complex preferences. A typical neural network can contain numerous nodes and even more inter-connections. The translation of a complex network should not result in numerous preferences relating to the same personalisable element, neither should it result in a very large preference with multiple nested IF statements. A typical user would find such a preference set impossible to understand even if it is human-readable.

Another challenge is allowing the user to view their entire preference set, not just the preferred preference outcomes for the context the user is currently in. This requirement indicates that an exhaustive rule search is required. Such searches are expensive both in terms of processing time and memory and their cost increases with the size of the network. It is undesirable for a long delay between the user requesting to view the preference set and it being displayed. Further, the mobile nature of pervasive devices places a limit on processing power and memory.

4 PRESENTING USER PREFERENCES TO THE USER

The Persist ANN is a binary neural network that takes real-world inputs regarding the user's context and the selected preference outcomes. It is a single layer network but is described in terms of two layers for clarity. The context layer represents the user's context and acts as an input layer. The outcome layer represents the user's selected preference outcomes and acts as both an input and output layer hence retrieval is possible during network learning. Figure 1 shows the high-level Persist ANN topology and input/output fields.

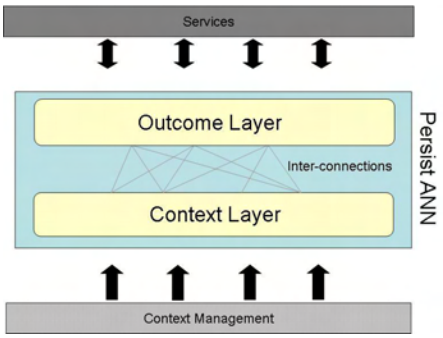

Figure 1. Persist ANN

The Persist ANN has been designed to overcome the challenges listed above. Binary activations and linear connections reduce internal complexity, hence simplifying the network interpretation process. The weight on each network connection is directly related to the strength of the association between some context node and some outcome node. Therefore, by comparing weights, weak connections can be disregarded. The benefits of such pruning have been recognised in rule extraction. It is generally agreed that in order to extract simpler, more concise rules, the network should have less connections. Therefore, by reducing the number of connections through the removal of weak, less influential ones we can more easily extract preferences that consist only of those context values with the strongest influence on the preference outcome. By reducing the number of connections we are also minimising the search space and hence the search time relieving the burden on processor time, processor power and memory required.

Consider the situation where we have the network state shown in Figure 2. (Each node is connected to every other node in the opposite layer. The strength of each connection is determined by the weight value at the synapse on each connection):

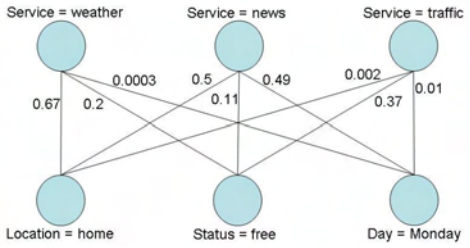

Figure 2. Example Network State

First we prune the network to remove the connections with a synapse value below some threshold Θ (e.g. 0.2). This means we loose connections 0.0003, 0.11, 0.002 and 0.01. From the remaining connections we can now create IF-THEN-ELSE preference rules. One possible approach involves systematically activating the possible patterns of context nodes and capturing the preferred outcomes for each pattern creating a set of IF-THEN preference parts. These individual preference parts can then be merged together using ELSE branches and logic simplifications. In this example the resulting IF-THEN-ELSE preference will be:

IF location='home' OR (location='home' AND status='free')
THEN
service='weather'
ELSE IF status='free'
THEN
service='traffic'
ELSE IF day='Monday'
THEN
service='news'

5 CAPTURING MANUAL USER UPDATES

Giving the user complete, final control of personalisation means allowing the user to manually manipulate their preference set. This could be creating new preferences, deleting preferences or changing the outcome or conditions of existing preferences. Whatever the case, the user updates must be captured in the human-readable format and translated into the internal preference format understood by the system (i.e. neural network weights).

As mentioned above, one of the rule extraction research goals is to use a rule base (usually rules extracted from the neural network) to create new neural models. In a sense it is translating rules back to neural network form. Some inspiration can be taken from this area to overcome the challenge of translating the user's preference set back into neural network form. However, this is limited as in our problem domain we do not wish to create a new network model. The Persist ANN will continue to exist internally after human-readable preferences have been extracted and presented to the user. If the user performs any updates on their preference set while in human-readable format we wish to translate only those updates back into the existing network. We do not wish to create a new ANN.

To create a new preference the user must specify an IF-THEN-ELSE Daidalos preference format rule through the preference GUI. Each IF-THEN branch will contain a context condition (set of context tuples e.g. 'location = home') and an outcome (e.g. 'service = weather'). To translate a new preference into the internal network structure we complete several steps. Firstly we must identify context nodes that represent the stated context condition values. Secondly we need to identify the outcome node that represents the related preference outcome. Finally we increase the weight value on the connections between these context and outcome nodes making the connections excitory. This ensures that the outcome node will fire when the user stated context condition is true. When the user wishes to delete a preference we prune the network to remove the output nodes that relate to the deleted preference. We cannot remove the nodes relating to the context values in the preference condition as these nodes also provide input to other output nodes.

When the user updates an existing preference they can either change the context condition or change the outcome. If the context condition has changed we must identify the context nodes that represent updated attributes in the context condition. By manipulating the weight value on the connections between such nodes and the node representing the preference outcome we can ensure that context nodes (and hence the related context values) reflect the user specified influence over the activation of the outcome node. If the preference outcome has changed we must identify the outcome node that represents the updated preference outcome. As above, we alter the weight value on the connections between this outcome node and the related context nodes to ensure that they are excitory or inhibitory as specified by the user.

Consider the example above. Imagine the user changes the IF-THEN-ELSE preference (generated from the network) to:

IF location='home' OR (location='home' AND status='free') OR day='Monday'
THEN
service='weather'
ELSE IF status='free'
THEN
service='traffic'

To translate this back into the network we must ensure that the strength of the connection between 'day=Monday' and 'service=news' does not over-ride the strength of the connection between 'day=Monday' and 'service=weather'. To do this we must inhibit the former connection and excite the latter. Figure 3 shows the updates represented in the network.

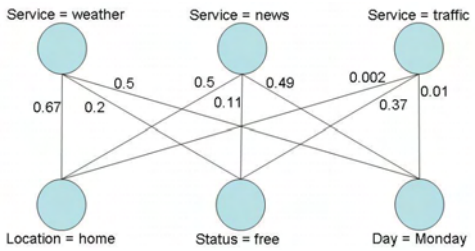

Figure 3. Updated Network State

Notice that the strength of the connection between 'day=Monday' and 'service=weather' has been increased just

enough to allow it to over-ride the association between 'day=Monday' and 'service=news'.

6 CONCLUSION

There is no doubt that personalisation mechanisms can greatly enhance user experience in a pervasive environment. They unburden the user of resource management tasks, adapting the environment to meet the current needs of the user. However, to provide accurate personalisation the system must maintain an accurate set of user preferences on which it can base decisions. Without such information it is impossible to know what actions will help rather than hinder the user.

There are two main approaches to the management of preferences. Explicit personalisation places the onus on the user to manually maintain their preference set through some GUI. Although this places the user in complete control, the burden of maintaining such a large rule base is great. This undermines the benefits of personalisation and can often lead to a sparse preference set and hence inaccurate personalisation. At the other extreme, implicit personalisation uses monitoring and learning techniques to maintain a preference set on behalf of the user. The benefit of such an approach is the minimal burden on the user however care must be taken to provide some method of user control. Without such functionality the user cannot alter system behaviour to reflect new situations or behaviours in a rapid way.

Therefore, the most successful systems take a hybrid approach providing implicit personalisation where possible but also providing a GUI through which the user can manually manipulate their preferences and take final control. The learning approaches employed by such systems often fall under two types; Rule-based learning algorithms (which store preferences as rules) and network algorithms (which store preferences in some network structure). Rule-based learning algorithms have the advantage that their output is easily translated into human-readable form allowing their knowledge to be understood. However for many problems they lack the success and flexibility of network approaches. The internal complexity of networks means they are very efficient learning tools for some problems but also means that often humans must accept their output without understanding why the network reached this conclusion.

The Persist project aims to gain the best of both worlds by utilising the Daidalos IF-THEN-ELSE rule-based learning techniques as well as introducing a more complex neural network learning approach that can be translated into IF-THEN-ELSE rule format for human understanding. This paper illustrated how the Persist personalisation system will accomplish this task. The Persist personalisation system will be fully demonstrated in 2010.

7 ACKNOWLEDGEMENT

This work was supported by the European Union under the FP7 programme (Persist project) which the authors gratefully acknowledge. The authors also wish to thank all colleagues in the Persist 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 Persist consortium. Apart from funding the Persist project, the European Commission has no responsibility for the content of this paper.

REFERENCES

[1] WEISER, M., The Computer for the 21st Century, Scientific America, 256(3), pp. 94-104, 1991.

[2] YOSHIHAMA, S., Chou, P., Wong, D., Managing Behaviour of Intelligent Environments, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom '03), pp. 330-337, 2003.

[3] LESSER, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., Zhang, S. XQ., The Intelligent Home Testbed, Proceedings of the Anatomy Control Software Workshop (Autonomous Agent Workshop), pp. 291-298, 1999.

[4] 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), pp. 328-340, 2006.

[5] ZIEBART, B.D., Roth, D., Campbell, R.H., Dey, A.K., Learning Automation Policies for Pervasive Computing Environments, Proceedings of the 2nd International Conference on Autonomic Computing (ICAC '05), pp. 193-203, 2005.

[6] SI, H., Kawahara, Y., Morikawa, H., Aoyama, T., A Stochastic Approach for Creating Context-Aware Services based on Context Histories in Smart Home, ECHISE2005, Pervasive 2005, pp. 37-41, 2005.

[7] CORDIER, C., Carrez, F., Van Kanenburg, H., Licciardi, C., Van der Meer, J., Spedalieri, A., Le Rouzic, J.P., Zoric, J., Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform, Workshop on Applications and Services in Wireless Networks (ASWN '06), 2006.

[8] MCBURNEY, S., Papadopoulou, E., Taylor, N., Williams, H., Adapting Pervasive Environments through Machine Learning and Dynamic Personalisation, Proceedings of the International Conference on Intelligent Pervasive Computing (IPC '08), pp. 395 - 402, 2008.

[9] CROTTY, M., Taylor, N., Williams, H., Frank, K., Roussaki, I., Roddy., M., A Pervasive Environment Based on Personal Self-Improving Smart Spaces, Workshop on Architectures and Platforms for AmI at the European Conference on Ambient Intelligence 2008 (AmI 08), 2008.

[10] TAYLOR, N., Personal espace and Personal Smart Spaces, 1st PerAda Workshop on Pervasive Adaptation at the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008), 2008.

[11] MCBURNEY, S., Papadopoulou, E., Taylor, N., Williams, H., Managing User Preferences for Personalisation in a Pervasive Service Environment, Proceedings of the 3rd Advanced International Conference on Telecommunications (AICT '07), pp. 32 - 37, 2007.

[12] QUINLAN, J.R., C4.5: Programs for Machine Learning, Morgan Kauffman, 1993.

Smart Space a new dimension of context

Luca Lamorte, Claudio Venezia

Abstract Telco Networks are rich contextual information containers and the exploitation of contextual information is expected to be a key success factor for the next generation Telco services. On the other hand the Internet of things is driving us towards a scenario in which network connected intelligent objects will be capable of offering services to users. We expect that those services and their availability will be part of the context itself.
Smart spaces will therefore compound context and available services.
In this article we describe how a context aware architecture has been evolved towards a novel context aware SOA which provides smart services according to users' context. This paper presents a real use case for defining the interactions among the different actors, a description of the architecture and some future works to define the next steps.

1 INTRODUCTION

Every year manufactures market a plethora of new gadgets, technologies and devices, improving their functionalities and design. These devices are network enabled and capable of sharing information with each other within an ad hoc network or through the Internet.
In the future we expect to be able to achieve ambient services which span over different devices and provide humans with innovative interaction means. What is really missing in this futuristic scenario is a reference framework able to translate and adapt the different languages they are speaking and to define a suitable way of discovering them when needed.
Today people are surrounded by these kind of internet of things, which are everywhere, which would be ready to be used if you were aware of how to invoke their services.
They might be triggered by an explicit request of the user, or suggested considering user context or preferences.
With the term user context we intend the collection of information related to the user which may come from the available collecting sensors. Adding to the various sensor based information about a user the list of ambient available services we got to our definition of Smart Context.
The Smart Context represents a user status but also the potential interactions he might enact with the surrounding digital intelligence.
Our platform is able to:

- collect information from all available sensors with respect to a particular user and determine her context
- collect information about the available service
- perform a correlation between available services and user context

Each service, correlated to a location or context, becomes part of the context itself. This opens up a new scenario, not only providing the user with the context sensible service when he/her needs it, but also to understand which ones he might want to use depending on his situation or kind of place and activity he's doing.
The actual research about user context aims at an exploitation of contextual information for customizing general purpose services according to the situation and expectation of the user in that context. But if he's surrounded by specific services and applications of any potential interest for him, this is a new way of pushing a new context to the user.

2 USE CASE: MULTIMODAL INTELLIGENT FRAME

Luca has invited Claudio in his high tech house, to see some pictures of Claudio's recent trip. Claudio carries his brand new mobile with all stored pictures. When Claudio enters Luca's house, a message is delivered to his mobile, triggering that Luca's house provides some interesting services that he can use.
Along with the available services list the message provides a link to reach a service console for using them. This message has been delivered by the Context Aware Platform [CAP], which has discovered Claudio in Luca's house.
Claudio, that is a curious guy, tries the link. A custom mobile web page, lists all the available services which Luca's house provides and acts as a remote controller. As first choice, the service console suggests Claudio to use Luca's network. Claudio was set by Luca as a friend in his contacts list, and special services, like fast and free connection are available for friends in his house.
A second service is the HD Television which provides a channel to show pictures. After selecting this service, the console offers to Claudio an interface to remotely handle the frame by changing the displayed picture via buttons like next, stop, previous or defining preferences like output, layout and finally selecting an input source. Claudio can use his mobile as a source and finally display to Luca those pictures. Moreover with the handling frame buttons, Claudio can easily stop the presentation by his mobile to better describe the trip dynamics.

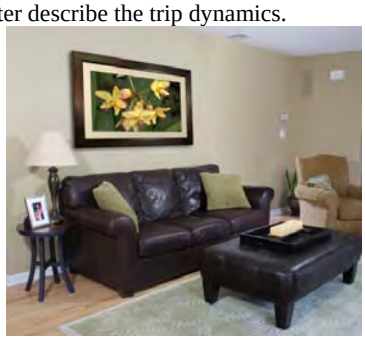

Figure 1. Luca's photo frame

3 Context Awareness Platform Architecture

In order to achieve the enablers for implementing a use case such as the one above we needed an architecture [6] capable of:

- collecting context information from different kinds of providers
- store contextual information and be able to aggregate them and provide to requestors

As said with we use the term context to indicate the collection of information available from user's surrounding environment, his/her terminal, network connectivity along with his/her profiles and preferences. It is a large amount of information which tends to grow proportionally to the observing time.
The logical process of context high-level abstraction and extraction or inherence is shown in Figure 2:

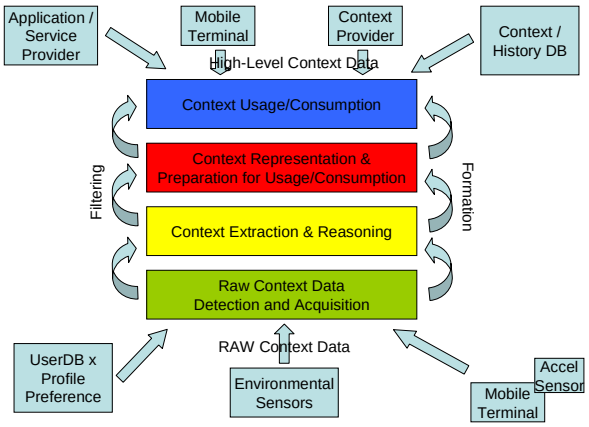

Figure 2: Context Processing

We've pursued a comprehensive and distributed context-aware system capable of aggregating and process a variety of context information. Despite that in context-aware system, domain knowledge is very much tied to applications; we resolved that building vertical solution was not effective. Therefore one of the goals of any context handling system is to split context and available applications. The Context Management Architecture shown in Figure 3 enables reuse and support for many applications from context acquisition to the context usage or consumption. This CMA designed according to the prosumer-consumer paradigm where some entities producing context are Context Providers (CP) while other entities consuming context are Context Consumers (CC). These entities are communicating each to other through a central entity named Context Broker (CB) which also provides some additional functions in the system.
The main characteristics of the entities shown in Figure 3 are the following:
Context Broker (CB) is the fundamental entity that creates and manages the relationship between CP and CC. The CB performs context source discovery and management and subject-based lookup service. Context Provider is the logical point where context is detected and acquired or extracted from other context sources is the Context Provider (CP). A context provider provides mechanisms for on-demand queries and may optionally support subscription based notifications based on defined event occurrences, e.g. general context change or timer expiration.

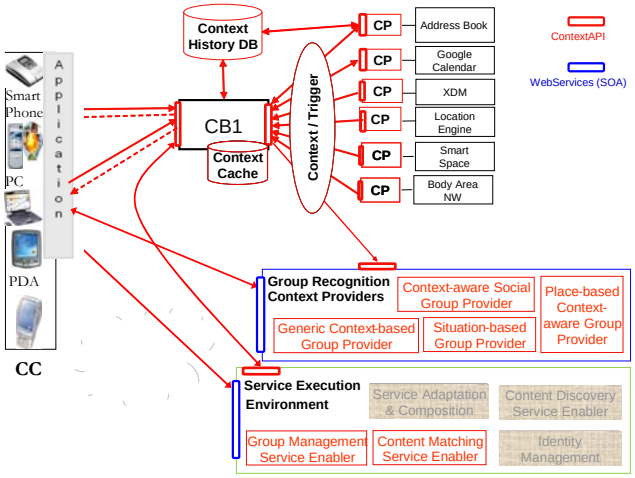

Figure 3 Context Management Architecture

Along with the acquisition of context data, the context provider performs data aggregation, fusion and inference. These activities are performed by different context providers, based on their individual internal logic and context output. A context provider is supposed to maintain context metadata and historical context. Within the CMA the context inference process is candidate to be embedded into CPs.
Reasoning and interpretation are seen as crucial tasks of the context-aware system, for deriving high-level contextual information from lower level context or raw data (e.g. databases or sensor output).
Most of the related research activities saw the context inference process as unidirectional process and rather static as far as the control or configuration are concerned. Due to the highly dynamic nature of mobile communication systems (e.g. mobility, activity change), reasoning and inference mechanisms need to be able to cope with the continuously change of contextual information;
Context Consumer is the logical point where context is used (or consumed) accordingly to context-aware service logic is the Context Consumer (CC) entity. The CC uses event-based model and uses query-based publish-subscribe mechanism for context data. In addition to context-aware applications, other architecture entities, like a CP or service enabler, can also assume the role of CC;
Service components or service enablers perform a set of generic control functions, used by applications and other service enablers, as illustrated in Figure 3.
These enablers usually act as a bridge between the CP and CC. Within the CMA, a service enabler assumes the role of CC and interacts with the CP. This initial interaction is always mediated by the CB.
Of course such architecture has a single point of failure represented in a Context Broker. However it has also many advantages respectively to other models based e.g. distributed or end-to-end paradigm. This model allows maintaining the control

over all the context information processed by the system and all the entities interconnected within the system and handling the context are connected over Context Broker. So the CB simplifies look-up based on required context for the entities within the system. Moreover, CB may have cache and history functions and then it may become the point of reference for already known and still valid context and for already expired context respectively.
The weak points of this solution may be easily solved by common practices applicable for robustness within centralized systems.

4 CONTEXT INFORMATION PROCESSING MODEL

Context information processing and the main CA components' relationships within this architecture is shown in Figure 4.

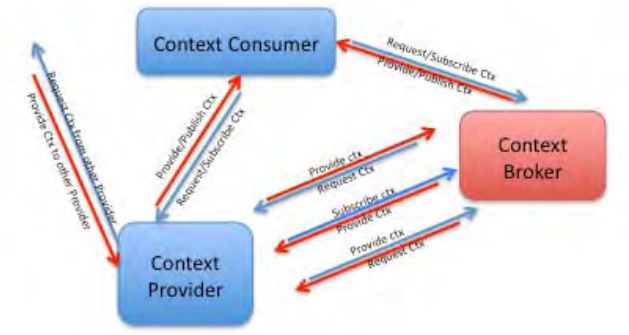

Figure 4: CMA entities relationships

Achieving a loosely coupled architecture was a critical requirement since the beginning.
Fixed Mobile Convergence of TLC and IP providers emphasizes the problem of heterogeneity and openness to 3rd parties applications using both types of networks. This issue is addressed within SOA concept [6] where enablers organize access to network's capabilities though standardized interfaces united within a service delivery platform, that may be opened by a service exposure layer towards 3rd parties applications respecting the identification, security and billing issues of external accesses in conventional systems.
The context management system has been created to avoid vertical context-aware applications and offer context-awareness to any application or service provider.
There is logic separation in context management, which remains within TLC domain, and application or service related functionalities provided by their respective providers and built on top of TLC features.
TLC features are abstracted in this case from service providers in a way that a service provider may not know how the functionality is implemented within the TLC domain.
This model creates a significant improvement in the business value creation chain; application logic inventors don't know all the particularities of TLC, while TLC are not always able to create services to attract users and create new revenue streams. The users take benefit of usage of heterogeneous network features without taking care to which networks they are connected to and which technology is used for data transmission.

In order to maximize interoperability between providers from different domains, a common language for user information representation has been defined: "ContextML", an XML-based language, which states a meta-model for the representation that all providers need to comply with, in order to register them with a User Information Broker and to enable potential Consumers to discover the user information they need.
For simplifying context management user's information have been subdivided by into scopes, namely sets related to the same information category. For example, the scope named "position" groups latitude, longitude and range with respect a certain entity's (e.g. user) location. Scopes can be atomic or aggregated, as union of different atomic scopes.
The ContextML schema is composed by the following elements:

- ctxEls: contains a set of information for a certain entity (response to the getContext method)
- ctxAdvs: contains the advertisement of provider features (providerAdvertising method) to the broker
- scopeEls: contains response from the Broker to the getAvailablesAtomicScopes method
- ctxPrvEls: contains response from the Broker to the getContextProviders method

Any user information given by a provider is characterized by an entity and a specific scope. When a provider is queried, it returns the required data in a XML document, which contains the following elements:

- contextProvider: a unique identifier for the provider of the data.
- entity: the identifier of the entity which the data are related to.
- scope: the scope which the context data belongs to.
- timestamp and expires: respectively, the time in which the response was created, and the expiration time of the data part.
- dataPart: part of the document which contains actual user information data which are represented by a list of a features and relative values through the <par> element ("parameter"). They can be grouped through the <parS> ("parameter struct") or <parA> ("parameter array") elements if necessary.

For example, the getCivilAddress method of the Location Provider for latitude 45.112 and Longitude 7.67 for user "luca" is invoked through the following HTTP GET:

http://svrXXX/LP/LocInfo/getCivilAddress?username=x&lat=45.112&lon=7.67

and returns the following XML content:
<?xml version="1.0" encoding="UTF-8" ?>
<contextML xmlns="http://ContextML/1.1"

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
  xsi:schemaLocation="http://svr/schemas/ContextML-1.1.xsd">
<ctxEls>
   <ctxEl>
       <contextProvider id="LP" v="1.0.2" />
       <entity id="luca" type="username" />
       <scope>civilAddress</scope>
       <timestamp>2007-01-24T16:05:19+01:00</timestamp>
      <expires>2007-01-24T17:05:19+01:00</expires>
      <dataPart>
            <parS n="civilAddress">
              <par n="street">Via Arrigo Olivetti 5</par>
             <par n="postalCode">10148</par>
             <par n="city">Torino</par>
             <par n="subdivision">TO</par>
            <par n="country">Italy</par>
           </parS>
      </dataPart>
</ctxEl>
```

Each specific domain of information is mapped by one or more scopes. As said a scope is a simple a priori aggregation of data with a semantic coherence, grouped together and identified by a name, which is used as parameter name (n attribute of <par> element) in ContextML. Such names could easily be mapped to concepts in an ontology.
The scopes currently defined are related to: location, calendar and device information.
In addition to the above We've added the list of the ambient services which would be triggerable by the user in a particular context (e.g. location plus activity and social information).
Basically ambient services publish their interfaces as contextual information. At any point in time a user can ask the platform if there is a smart service he can use for the time being.
In the case of our use case when Claudio approaches Luca's television he is notified of its availability and provided with the information for using it.
In fact the photo frame declares several interaction interfaces in the form of URI like next, stop, refresh, set interval, choose playlist.
The system stores these information and associate them to a physical location (e.g. room which hosts the photoframe) and/or a suitable fruition context (e.g user's is in leisure status).
Most of portable devices can be located via cell triangulation, Global Positioning System (GPS) or bluetooth and when a user approaches the photo frame the presence of the photo frame becomes actually part of his context. The contextML document will provide the following in addition:

```
<contextML>
  <ctxEls>
     <ctxEl>
       <contextProvider id="AmbientServProv" v="1.0.2" />
       <name>photoframe</name>
       <op>Http://smartspaceIP/frame/Home</op>
       <op>Http://smartspaceIP/frame/upload</op>
       <op>Http://smartspaceIP/frame/next</op>
       <op>Http://smartspaceIP/frame/stop</op>
       <op>Http://smartspaceIP/frame/refresh</op>
       <op>Http://smartspaceIP/frame/setInterval</op>
       <op>Http://smartspaceIP/frame/changePlayList</op>
       </ambService>
     </ctxEl>
  </ctxEls>
</contextML>
```

The URIs above points to the functionalities implementations which are exposed via REST interfaces, and available for use, if the requestor has the right privileges. Whether these REST functionalities are implemented as a Web application which resides on the photoframe or a network Web application which than instructs the photoframe via proprietary protocols is just an implementation detail which depends on the device capabilities.

5 SMART CONTEXT CLIENT GUI AND IMPLEMENTATION

From the really beginning of this work We had clearly in mind that users should have been the primary actors of this user centric contextual services ecosystem.
Given the importance of the mobility aspects it was a requirement to provide mobile friendly interfaces to this ecosystem. Basically the requirements are:

- mobile enabled experience
- platform independence
- a basic knowledge of the mobile capabilities [9]

The choice was then to achieve a mobile web based client and provide a mobile Web application. Users interact with a Web based Mobile Desktop. This mobile desktop periodically reloads as users change location and displays different services and information accordingly. Those data could also be overridden manually by users by expressing explicit preferences. This is a very dynamic phase of the scenario, because as the user moves around the mobile application changes.
The photo frame is a static element in the scenario, because its services, its location and space is fixed. The only thing that it has to do it is to announce itself as a service available for a context/space. This is done by sending its registration the the CAP as a context provider.
The Web based mobile desktop periodically asks a context broker for updated context information (receiving back a XML such as the one described in the previous chapter) and displays to the user the information about potential available surrounding services. The process is more complex because before sending back the XML result, the CAP detects the user identity, and the requesting client profile. Matching and crossing the information stored in the user profile, his/her preferences and context, it creates a customized result to fit that specific user and his/her situation. The page displayed on the mobile web desktop after the update will reflect exactly the context, services and the user's preferences.
When a user gets in proximity of a photo frame the mobile desktop GUI shows a section which displays the interaction possibilities. Basically several available remote commands are executable via HTTP/REST interfaces.
For what concern privacy and trust issues in our implementation the requirement for being able to interact with the photo frame is being part of the owner's social network, which automatically grants the right to use it. That's why Claudio's Web Mobile

desktop notifies him of Luca's photo frame presence and Claudio can choose to use this service. If the web mobile desktop is not running, CAP can notify the user in different ways according to the mobile device capabilities (MMS, SMS, Instant Message).
When notified Claudio can access simply clicking the FRAME button of his Mobile Web page, entering a page dedicated to the frame interaction where he can find functionalities like those listed below:

- more details about the frame like size, number of pictures available, refresh frequency, layout type, …
- changing the output layout, if the user has the authority. Some example can be: slideshow, split diplay by 2,4,6 …
- Handle the display experence using those operation:
    - auto
    - next
    - previous
    - platy/stop.
    - More/less info

Figure 6 Smart Context client

6 RELATED WORK

There is a lot of work related to SOA and context management. Both the i-Room [1] and Gaia [2] present smart office application scenarios. i-Room focuses on human computer interaction (HCI) in a single interactive meeting room. Gaia defines Active Spaces as physical spaces coordinated by a responsive context-based infrastructure. This infrastructure is made available to service applications by means of an operating system (Gaia OS), which provides context and event management services to running programs. As a future work, the Gaia team plan to federate Gaia Services to aggregate different active spaces. Cooltown [3] uses the technologies behind the Web to provide pervasive nomadic computing in urban environments. In Cooltown, interest places and resources are tagged with URLs or other identifiers that can be retrieved by users' personal devices by means of bar codes, RFIDs or IR transceivers. URLs may be used to access the different services related to their associated points of interest, and other identifiers may be resolved to URLs which link to the services related to the identified item. Resources are grouped in places, and for each defined place there is a place manager, which maintains directories of resources, acting both as a resolver for looking up resources in that place from their identifier and as a Web server providing information about resources. COBRA [4] takes advantage of multiagent systems to develop context-aware applications. It is based on a brokercentric architecture used to provide runtime support for context awareness in an Intelligent Meeting Room. In COBRA, the environment is divided in domains, and there is a broker for each domain, which is an autonomous agent that manages and controls the context model of the specific domain. Though COBRA brokers are intended mainly for context sharing, its centralized approach to management in each domain and the possibility of sharing context between domains through context federation is closely related to the approach we have taken to develop our hierarchical architecture for smart spaces

7 CONCLUSIONS & FUTURE WORK

Ambient intelligence and context awareness seem to be the ideal ground for the exploitation of the already consolidated SOA principles.
In the meantime, interoperability, lack of standards, and privacy are still important challenges to be mastered.
For the future our idea is evolving this architecture towards an "increasing" social awareness. Ambient services are often delivered to groups or communities of people. If the communication style is changing from p2p to group communication thanks to Social Networking applications we could expect that this might also apply to the service experience. For people belonging to the same community, who also share the same physical spaces, could be indeed fascinating to create and provide ambient services to their friends. Telecom Italia is participating to PERSIST exactly for the sake of researching on Smart Spaces technologies and pursuing a platform for achieving highly flexible and interoperable Personal Smart Spaces.

REFERENCES

[1] Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: Experiences with ubiquitous computing rooms. IEEE Pervasive Computing (2002) 67–74
[2] 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 (2002) 74–83
[3] Kindberg, T., Barton, J.: A web-based nomadic computing system. Computer Networks 35 (2001) 443–456
[4] Chen, H.: An Intelligent Broker Architecture for Pervasive Context-Aware Systems". PhD thesis, University of Maryland, Baltimore
[5] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an Ambient Intelligence Environment Using Embedded Agents. *IEEE Intelligent Systems*, 19:12-20, (2004).
[6] L. Lamorte, C.A. Licciardi, M. Marengo, A. Salmeri, P. Mohr, G. Raffa, L. Roffia, M. Pettinari, T.S. Cinotti, "A platform for enabling context aware telecommunication services," in: Proc. Third Workshop on Context Awareness for Proactive Systems, Guildford, UK (2007)
[7] C.Venezia, C.A. Licciardi, A. Salmeri, L.Buriano; "Rule based dynamic adaptation of mobile services based on context"; in the proceedings of ICIN 2008.
[8] W3C Ubiquitous Web Applications. http://www.w3.org/2007/uwa/ [Online; accessed 10-Feb-2007].
[9] W3C delivery context ontology. http://www.w3.org/TR/dcontology/ [Online; accessed 10-Feb-2007].

How to make Personal Smart Spaces Context-aware

Ioanna Roussaki[1], Nicolas Liampotis[1], Nikos Kalatzis[1], Korbinian Frank[2] and Patrick Hayden[3]

1 INTRODUCTION

Pervasive computing [1] or as otherwise called Ambient Intelligence [2] aims to assist users in their everyday tasks in a seamless unobtrusive manner. In this framework, there have been various research initiatives aiming towards the design and realization of smart spaces [3] in homes, offices, universities, schools, hospitals, hotels, museums, and other private or public places, where various automation facilities support the users. In these cases, research has focused on developing techniques to support building automation (such as intelligent light controls, window shutters, security systems, electrical appliances, door control, etc.), as well as mechanisms to adapt the behaviour of electronic devices (such as TV, radio and multimedia players), desktops and peripherals, etc. Nevertheless, these are fixed spaces that provide pervasive features and adapt to the user needs in a static and geographically limited environment.

However, mobile users require the same pervasive services wherever they are and whatever devices they carry along. Irrespective of the user's location, such a mobile pervasive system would be expected to provide access to devices and services in the user's current environment. Fixed smart spaces do not address the needs of mobile users, as outside their boundaries, only limited pervasive computing features are offered. For example, a Smart Home may control the devices within it and the services it offers to its residents, but it cannot easily share these with the mobile network of any visitor. To bridge this gap, the notion of self-improving Personal Smart Spaces has been introduced [4].

Personal Smart Spaces (PSSs) aim to couple the facilities offered by next generation mobile communications with the features provided by the static smart spaces to support a more ubiquitous and personalised smart space that is able to follow the user wherever he/she goes. As the owner of the PSS moves to different locations and places, his/her PSS interact with other mobile or fixed PSSs located in the owner's surrounding environment, aiming for a unique support for pervasive service provisioning. In a nutshell, a Personal Smart Space can be defined as a set of services within a dynamic space of connectable devices, where the set of services are owned, controlled, or administered by a single user. It facilitates interactions with other PSSs, it is self-improving and is capable of pro-active behaviour. Thus, a PSS is user centric and controlled by a single user, it is mobile (at least from user perspective), it allows for interactions with other PSSs and is capable of self-improvement and proactivity.

An inherent feature of PSSs is context awareness [5]. Context can be defined as basically all information that is relevant to a human-computer interaction [6]. However, even though context information holds out the prospect of enhancing user experience and increasing revenues; collecting it from various heterogeneous sources, disseminating it across distributed nodes, processing and managing it efficiently are not straightforward. Other context related features that need to be supported are: real-time control and management of the context sources; distribution of context information over heterogeneous networks and devices; inference of future or currently unavailable high level context information from raw context data based on various learning techniques; modelling, management, storage and processing of history of context data; support for privacy-aware access control facilities and secure distributed context event management mechanisms; etc. This paper elaborates on the mechanisms that have been designed in order to address the advanced requirements of PSSs regarding the establishment of a robust distributed context management framework.

More specifically, this paper is structured as follows. In Section 2, the context-awareness requirements that need to be addressed in PSSs are presented. In Section 3, an illustrative use case scenario is described, where the context-awareness aspects of PSSs are highlighted. Then, in Section 4, the context model that has been designed to represent the context information necessary to adapt the services provided by PSSs is briefly presented. Section 5 elaborates on the context management architecture that has been built to address the needs of PSSs. In this respect, the functional components of the context management system are presented, while emphasis is given to the mechanisms that support distributed context management facilities. Finally, in Section 6, the paper conclusions are drawn.

2 CONTEXT-AWARENESS REQUIREMENTS IN PERSONAL SMART SPACES

As already stated, a Personal Smart Space is a set of services that are owned, controlled, or administered by a single user, that are located within a dynamic space of connectable devices and that are capable of self-improvement and of demonstrating pro-active behaviour. This leads to numerous context-awareness requirements that need to be addressed in PSSs. A summary of these context-related requirements is provided hereafter.

- *Efficient Context Modelling and Semantics*, in order to represent the entire set of context data (both dynamic such as location, and static such as preferences) that need to be monitored, collected, stored and utilised in PSSs, along with the necessary metadata. More details on this are provided in Section 4.

[1] School of Electrical and Computer Engineering, National Technical University of Athens, Greece. Email: {**nanario**, **nliam**, **nikosk**}**@telecom.ntua.gr**.

[2] German Aerospace Center (DLR), Institute of Communications and Navigation, Germany. Email: **korbinian.frank@dlr.de**.

[3] Telecommunications Software & Systems Group (TSSG), Waterford Institute of Technology, Ireland. Email: **phayden@tssg.org**.

- *Distributed Context Management*, including distributed context maintenance, access, update and synchronisation, as well as provision of a transparent interface to context management, support of ad-hoc context exchange, real-time and non-real-time context handling. More details on this are provided in Sections 5.2.4 and 5.3.
- *Context Query Support*. Various context queries need to be supported by PSSs, such as identity-based (or navigational) queries, location-based, semantic-based, and time-based. More details on this are provided in Section 5.2.1.
- *Context Source Management*, including sensor data aggregation, context-source discovery, (de-) registration, configuration, etc. More details on this are provided in Section 5.2.5.
- *Context Source Management*, including sensor data aggregation, context-source discovery, (de-) registration, configuration, etc. More details on this are provided in Section 5.2.5.
- *Preference handling facilities*, in addition to the other context management facilities aforementioned, i.e. preference analysis, preference evaluation, and preference condition monitoring. More details on this aspect are provided in [7].
- *History of Context Modelling and Management* in order to support history-based context inference and access to past context information. In this respect, the user behaviour & status will also be modelled and recorded. More details on this aspect are provided in Section 5.2.3.
- *Context Event Management*, including support for distributed context event creation and propagation.
- *Context Inference*, i.e. extraction of high level context information from raw context data. In this respect, learning of context inference rules (CIR) needs to be supported, as well as preference learning algorithms, CIR individualisation, context association & pattern extraction/matching, and CIR learning from group knowledge. More details on this are provided in Section 5.2.2 and [8].
- *Group context*, i.e. context information of group of persons/users, including group preferences. More specifically, the following need to be addressed: efficient group context modelling and representation, group context management & maintenance, group context estimation & inference, context prioritisation & assessment for resource sharing and context/preference conflict resolution. More details on this are provided in [4].
- *Context privacy & security*. In this respect, the following need to be supported: access control over individual and group context; context integrity, reliability, confidentiality and availability; context-based access control; privacy policy learning; etc. More details on this are provided in [4].
- *Quality of context* modelling, management and exploitation, including soft context and uncertainty in context values and context inference rules.
- *Context sensitivity*, i.e. ability to support adaptation of the provided services to the context of their users.

In the scenario presented in the following section, several of the requirements above are addressed by the Personal Smart Spaces involved.

3 AN EXAMPLE SCENARIO

In this section we will present a scenario that demonstrates the possibilities of Personal Smart Spaces and explains the involvement of context in it. We assume that all actors carry smart phones with them that constitute their PSSs and coordinate communication with all devices in and outside this PSS. Also infrastructure provides services and context information via fixed smart spaces that interact with the actors' PSSs.

Tom, Steve, Susanna and a couple of other researchers have arranged to meet for an international project meeting. Tom is the host, while the other participants fly in and stay in a hotel that Tom has organized close to the meeting venue.

3.1 Scenario description

Steve and Susanna arrive in the airport at the same time. Their buddy finder application informs them, so they can meet and share the trip to the meeting. Right after landing, keeping track of her agenda, Susanna's PSS had identified that they are late and proactively cancelled the reservation at the car rental service and ordered a taxi instead. She offers to Steve to share the taxi.

The taxi's PSS is responsible for adjusting the seats, so that Steve and Susanna are as comfortable as possible given their height and size. The taxi driver is already informed about the meeting venue, the price is agreed by the PSSs and they arrive on time.

When the first guests are about to arrive, Tom is notified at his office PC. He heads to the meeting room. As he enters the climate controls are set according to his preferences. As the weather is fine and warm, the windows are opened automatically to let in the fresh May morning air.

When Steve and Susanna enter the building, a shared display in the foyer welcomes them and informs them of the day ahead. Their PSSs had interacted with the display's PSS and selected the appropriate information for them. Furthermore on their smart phones they are offered the possibility to check the current activity in the meeting room and to check the availability of the remaining colleagues from the project.

While Steve usually wants to glance at the menu at the canteen and the weather report for the rest of the day, if he is in his office and therefore interacts with his own company's PSS, this information is not applied now when he is on a business trip.

Susanna and Steve are guided to the right meeting room. There they meet Tom and a new colleague, Joan. Joan talks to Susanna for a while and their PSSs interact – each infers that they are talking to each other and that they have not met before – at least their PSSs haven't! Susanna takes her SmartPhone from her pocket and sees the proactive prompt that just needs her confirmation to exchange electronic business cards with Joan.

The meeting starts and after some words of introduction by Tom, Steve is the first to present his work. He walks to the front of the room and given the agenda and behaviour, his PSS infers that he is about to start the activity "presenting" and starts interaction with the wall display's PSS. He is shown a context-aware selection of suitable documents – related by last-use and association with the calendar entry – and hits the icon for the

slides he prepared. The presentation is started and also logged in the automated meeting minutes that tracks the presentations shown and the speakers. An incoming phone call from his Yoga friend Katie is routed to his voice mail as he is busy presenting and the call urgency was indicated by Katie to be rather low.

Susanna is impressed by Steve's research results. She accesses the presentation from the common automated meeting minutes and wants to print it. Her PSS locates the closest printer, sends the print job, and guides her there automatically.

When the meeting approaches its end for the day, the PSSs interact to select a restaurant for dinner, taking into account the location, the weather and of course the participants' preferences regarding food and dinner time.

Having agreed on a restaurant in the city centre, Steve decides to drop his baggage at the hotel first. Therefore he does not join his colleagues who go the centre immediately, but decides to follow them later by public transport.

Steve's PSS selects the most convenient train to the city centre for him and interacts with the transport management system to acquire a ticket. Finally in the train he falls asleep, as he is tired after the hard working day, but his context aware alarm wakes him up just in time before he reaches the station where he has to get off.

Having enjoyed a wonderful dinner with his colleagues, they head together to their hotel in Joan's rental car. The car's navigation system guides them automatically to their hotel taking into account the broadcasted traffic situation.

3.2 Scenario analysis

This scenario shows a wide range of applications for PSSs and context. Context is used:

(1) in the process of service selection
(2) for service configuration
(3) during service execution
(4) to determine current preferences
(5) to proactively start services

Many different applications and basic services are used by Steve and his colleagues in this scenario, though only some crucial ones can be discussed in detail:

Firstly the car adaptation system, provided by the car PSS is of interest. It starts proactively (5) and incorporates the persons' preferences (4) and context during its execution (3). The relevant context information is height, size and weight of the passengers. Obviously, also the precise location of the passenger is needed, even within the car, and finally the user intent to enter the car resulting from location, movement and future calendar entries is responsible to start the service.

The room adaptation system (3, 4, 5) (as well as the restaurant finder service) furthermore includes weather and temperature information next to personal preferences.

In the call redirection application (2, 4, 5) mainly activity is the relevant context information. It depends on preferences, time, calendar entries, location, movements/status (like standing, walking, sitting and others) and the available as well as the currently used services.

The printing sub-scenario illustrates among others context aware service selection (1), based on features of the available printing services, but also on your current indoor position and proximity (including walking restrictions like walls etc.)

Finally the context aware navigation service or the context aware alarm (3) have to be aware of the current time, the user's target coming calendar entries as well as the map of the relevant area. Monitoring the current activity (walking-direction, but also sleeping) and precise location it calculates the best route to the target respectively initiates the alarm.

4 CONTEXT MODELLING

In order to efficiently represent context information in a PSS environment, the Persist Context Model (PCM) has been developed. The PCM includes all the classes that model the context information to be retrieved, exchanged, maintained and managed in general in the PSS. The scope of the proposed context model is to represent necessary information in an appropriate and uniform way for all the functional components of the PSS framework. The basic informational concepts used by the context model are the data classes: `CtxEntity`, `CtxAttribute`, and `CtxAssociation.` In addition to these core classes, there is the `CtxIdentifier` class that mainly addresses internal context management requirements, as well as the `CtxQualityAttribute` class, which further augments the model with Quality of Context elements.

The core concept upon which the context model is built is the *Entity*. An *Entity* corresponds to an object of the physical or conceptual world. For example an *Entity* could be a person, a device, or a service. The *Attribute* class is used in order to describe an *Entity*'s properties. To this end, many *Attribute* classes might be assigned to an *Entity*. Concepts such as the name, the age, and the location of a person are described by different attributes. Similarly, attributes describing a device's properties might be the identity, the voltage, and the operational status of the device. In a nutshell, the `CtxAttribute` class identifies an entity's status in terms of its static and dynamic properties and therefore, it captures all context information items that are used to characterise the situation of the owner entity. The `CtxQualityAttribute` class provides Quality of Context meta-data to Attributes [9]. Thus, each Attribute may be accompanied by an instance of the `CtxQualityAttribute` class. Examples of the Quality of Context properties provided by this class are: probability of correctness, frequency, precision, price, timeliness, etc.

Relations that may exist among different entities are described by the `Associaton` class. We identify two ways of associating entities. A peer relation among entities is described by the concept of the *undirected association*, while a non-peer relation among entities is described by the concept of *directed association.* In the latter, only one of the associated entities can be the originating point and is then called parent entity, while many entities could be the target points and are then called child entities. On the other hand, the undirected association represents relations among peer entities, where there is no owner or parent entity. Example types of directed associations are: "*owns*", "*uses*", "*locatedIn*", while types of non-directed associations are: "*friends*", "*schoolmates*", "*fellow passengers*", "*colleagues*", etc.

In a nutshell, the proposed context model is built to describe the situation of "*a person owning a device*" as follows: "The `EntityA` of type `Person` carries an `Association` of type `Owns` that connects it with `EntityB` of type `Device`. For this directed association, `EntityA` is the parent entity, while `EntityB` is the child entity.

The aforementioned `CtxIdentifier` class contains all the information that is necessary to uniquely identify the context information items and it is assigned to all entity, attribute and association instances. The string representation of the `CtxIdentifier` can be used by any PSS enabled context management system in order to retrieve the identified context information. The format of this identifier is as follows:

`CtxId: PSSid/DevID/ModType/Type/Number`

The individual parameters involved are described hereafter:

`Pssid`: A unique identifier of the PSS where context information was first stored.

`DevID`: An identifier of the device where the respective context information was initially sensed/collected and stored. This is the home device ID, the role of which is described in Section 5.3.

`ModType`: Describes the type of the context model object, i.e. is one of the following: Entity, Attribute, or Association.

`Type`: A semantic tag that characterizes the context model object.

`Number`: A unique number within a single PSS.

As already described, a PSS may include various devices, on each of which a context management system resides. In a system such as this, the wealth and heterogeneity of context data strongly discourages the adoption of a centralised or completely distributed context data management scheme. For this reason and as it will be further described in Section 5.3, we have introduced a *"Hybrid distributed-centralised context management"* approach for managing and storing context data. In order to achieve this, we use specific flags indicating whether changes on a context model object should be forwarded to other context management systems of a PSS or not.

To address the PSS requirements regarding context semantics, a semantic taxonomy has been introduced that includes the various context types as tags and dictates how these various context tags can be combined. A segment of this taxonomy is depicted in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ModelObjects>
  - <Entity type="PERSON">
      <Attribute type="NAME" />
      <Attribute type="LOCATION" />
      <Attribute type="AGE" />
    </Entity>
  - <Entity type="BUILDING">
      <Attribute type="NAME" />
      <Attribute type="LOCATION" />
    </Entity>
  - <Entity type="OFFICE">
      <Attribute type="NAME" />
      <Attribute type="LOCATION" />
    </Entity>
  - <Entity type="MANAGER">
      <Attribute type="NAME" />
      <Attribute type="LOCATION" />
      <Attribute type="STATUS" />
    </Entity>
  - <Entity type="SERVICE">
      <Attribute type="NAME" />
      <Attribute type="LOCATION" />
      <Attribute type="PRICE" />
    </Entity>
  - <Entity type="DEVICE">
      <Attribute type="NAME" />
    </Entity>
  - <Association type="HASSERVICEPREFERENCE
      <Entity type="SERVICEPREFERENCES" />
      <ParentEntity type="PERSON" />
    </Association>
  - <Association type="ISLOCATEDIN">
      <Entity type="PLACE" />
      <ParentEntity type="PHYSICALOBJECT" />
    </Association>
  - <Association type="ISSTUDENTOF">
      <Entity type="PERSON" />
      <ParentEntity type="PERSON" />
    </Association>
  - <Association type="AREFRIENDS">
      <Entity type="PERSON" />
    </Association>
</ModelObjects>
```

Figure 1. A segment of the context semantics taxonomy for Entities, Attributes and Associations

Finally, it should be mentioned that all context model objects are marked with a timestamp indicating their most recent update time.

5 CONTEXT MANAGEMENT ARCHITECTURE

The efficient management of context information is central in pervasive computing. Here, we present the Context Management (CM) framework of Persist which was designed based on the context model presented in Section 4 in order to cover the context-awareness aspects of Personal Smart Spaces. More specifically, this section is structured as follows. Subsection 5.1, provides a high-level view of the CM architecture. Subsequently, in Subsection 5.2, the functional components comprising this architecture are described, while Subsection 5.3, presents our approach for scalable distribution of context information among PSS devices.

5.1 High-level architecture

As illustrated in Figure 1, the CM framework acts as an intermediate layer between PSS/3rd party context-aware services and the sources of context information. This figure also provides a high-level view of the CM architecture introducing functional components, as well as, their interdependencies.

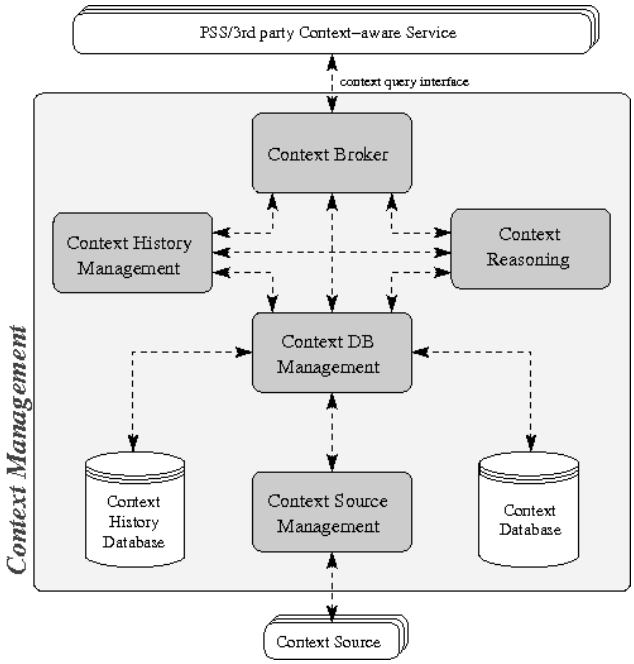

Figure 2. High-level Context Management architecture

An outline of the identified components follows, while a more detailed description is provided in Subsection 5.2:

- *Context Broker*: Provides context consumers with a query interface for retrieving, adding, removing, and updating context data.
- *Context Reasoning*: Uses probabilistic methods in order to derive high-level context information based on raw sensor data and/or context history.
- *Context History Management*: Collects, maintains and processes historic context data.

- *Context DB Management*: Translates context queries into standard SQL queries which are then executed in the underlying databases.
- *Context Source Management*: Manages context sources and collects their information.

Apart from the components above, the CM architecture comprises a *Database Management System* (DBMS) which enables access to the actual context repositories, i.e. the *Context Database* and the *Context History Database*. The former database is used for current context information, while the latter stores past data. Both database schemata conform to the context model described in Section 4.

Any off-the-shelf relational database meets the requirements of this model, however, our framework considers different DBMS solutions depending on device capabilities. In this context, a fully-featured DBMS, such as MySQL, can be deployed on a resource-rich PSS device, while a less-featured DBMS, like Apache Derby or SQLite for example, is more suitable for resource-constrained devices, thus, supporting current mobile smart-phone technology.

5.2 Functional components

The functionality of the components comprising the CM architecture is described in the subsections that follow.

5.2.1 Context Broker

The Context Broker manages access to context information. More specifically, it supports queries for retrieving, updating, adding or removing context data. Context retrieval queries in particular, can be performed both *synchronously* and *asynchronously*. For the latter case, context consumers are required to subscribe for context update notifications through the PSS Event Management component [4].

Context queries can be classified as follows:

- *ID-based*: As already described in Section 4, Entities, Attributes and Associations, which constitute the building blocks of context information, are uniquely identified by their Context Identifier (CID). Context consumers can specify CIDs in their context queries in order to identify the data items they are interested in.
- *Location-based*: This is actually a special use case for id-based queries and allows for discovering new context information based on location hierarchies modelled by Entities and their Associations. For example, given the CID of a place Entity, the "isLocatedIn" Association can be followed to discover the Entities located in that particular place.
- *Semantic-based*: When the CID is not known beforehand, context consumers can specify semantic criteria, i.e. tags, in their queries. The Context Broker uses these tags for best-match retrieval from a taxonomy of semantic tags with which context data items may be associated [10].
- *Temporal-based*: This type of query provides context consumers with access to past, as well as, future, i.e. predicted, values of context data. The actual processing of such data is managed by the Context History Management component which is described in Subsection 5.2.3.

Apart from context query handling, the Context Broker enables inter-communication among the CM subsystems of PSS devices.

To further elaborate the functionality of the Context Broker, we describe the processing of an ID-based context query. First, the Context Broker examines the PSS identifier which is encapsulated in the CID associated with the query in question. Based on this identifier, it can determine whether the relevant context data item is associated with a remote or the local PSS. For the former case, the original query is forwarded to the Context Broker of an available device of the identified PSS, while, for the latter case, the local context repository is checked. In case of an empty result from the local database, the Context Broker attempts to forward the context query to the *master* device of its PSS and, if that is not available, to the *home* one (see Subsection 5.3 for a definition of these terms). As a "last resort", if the requested context information is not available, either locally, or remotely, the Context Broker can invoke the Context Reasoning component in order to infer this information.

5.2.2 Context Reasoning

Some pieces of context information are directly derived from sensor readings. Location, for instance, can be determined from a number of sensors, such as GPS receivers or RFID readers. However, other pieces of context information, like "user activity" or "busy status" for example, cannot be assessed in a straightforward way. This so called high-level context information can be derived from the Context Reasoning component which provides a probabilistic *context inference engine*. The functionality of this engine is twofold:

1. To extract context estimation rules, i.e. inference rules, based on context history.

2. To infer high-level context based on the relative inference rules and raw sensor data.

Given the high temporal constraints of context-aware services and the computational complexity which is inherent in most probabilistic methods, time efficiency is a key factor in context inference. The Context Reasoning component addresses this issue through the use of *bayeslets* [11], which allow for prompt delivery of otherwise unavailable context data, as well as, refinement of existing, yet inaccurate information.

5.2.3 Context History Management

The Context History Management component is responsible for the collection and processing of *History of Context* (HoC) [12]. Maintaining this data is of great importance to the CM framework of Persist as it supports:

1. Inference of *current* context information, which is no longer available, based on HoC.
2. Prediction of *future* context information based on periodic context data patterns extracted from HoC.

It should be pointed out that this component does not automatically monitor all types of context information for inclusion in the history database. Instead, the PSS owner has to explicitly register context types for HoC maintenance. Registration ensures that whenever these context types are updated in the (current) context database, the Context History Management component is notified and appropriately updates the context history database after processing and scoring the relative data items, evaluating the likelihood of their occurrence. Based on these updates, time-dependent attenuation can be

applied to all past context values in order to extract context prediction rules.

5.2.4 Context DB Management

As already stated, the CM framework comprises two databases for storing context information, one for current values and the other for historic data. Hence the need for an intermediate layer between the context query interface provided by the Context Broker, and the underlying DBMS that actually controls the storage, management, and retrieval of data in these databases. The functionality of this layer is implemented by the Context DB Management component which is able to translate context queries submitted by PSS/3rd party context-aware services into standard SQL queries that can be executed in the framework's DBMS.

Another responsibility of the Context DB Management component is to provide the Context Source Manager with an interface to manage the QoC meta-data associated with context information, i.e. to assign and update the relative QoC attributes.

5.2.5 Context Source Management

The Context Source Manager component manages communication with diverse context sources. More specifically, it is responsible for discovering, (de-)registering, configuring and collecting context information from available sources. In the presence of multiple sources for the same piece of context information, this component is able to select the appropriate source based on the QoC requirements of context-aware services. Re-configuration of registered context sources may also be invoked based on such requirements. For example, if a context-aware service requires more frequent updates on location information, the Context Source Manager can dynamically re-adjust the sample rate of the location sensor. It should be noted that the overhead in resource consumption of both the sensor and the PSS device is taken into account for this process.

5.3 Distributed context management

This subsection deals with context management across the device nodes of personal smart spaces.More specifically, it describes a scalable scheme for distributing context data among multiple devices forming a single PSS. The following distribution approaches have been considered:

1. *Centralised context management*: The context database is hosted by one device per PSS. All context queries must be forwarded to that particular device for processing.
2. *Fully distributed context management*: Each device in a PSS hosts a copy of the context database. Context information is replicated in all devices, thus, context queries can be handled by any of them.
3. *Hybrid distributed-centralised context management*: Each device in a PSS hosts a context database; yet, context information is not replicated in all of them. There is, however, one device whose database contains every piece of context information and is able to handle all context queries. Regarding the other devices in the PSS, context queries cannot always he handled locally and must thus be forwarded to the appropriate device for further processing.

The first two approaches would be sufficient for fixed PSSs, however, scalability, in terms of processing, storage, remote communication and, power consumption, would be an issue for PSSs where both fixed and mobile node devices are involved. Therefore, we consider the hybrid distributed-centralised approach as more appropriate for Persist.

In the hybrid approach, all devices within a PSS contribute to collecting context information and are assigned different roles based on their capabilities. The following roles have been distinguished:

- *Master device*: A single device within a PSS, usually one with server capabilities, i.e. high processing power and storage capacity, is elected as the master device. It aggregates context information from all devices of the PSS and is responsible for maintaining HoC data in order to support inference of current and future context. Being the core of the CM framework, this device is intended to be always on.
- *Slave device*: Apart from the master device, every PSS may have one or more slave devices which contribute to the collection of contextual data. Slave devices periodically send collected data to the master one. If the master device becomes unavailable, a slave one may be elected in replacement.
- *Home device*: A device is considered home for a particular piece of context information if the latter is relevant to this device only.

For example, Attributes which are derived from context sources attached to a PSS device, have that particular device as their home one.

Nevertheless, it should be highlighted that static context information, as defined in 4, is fully distributed in the hybrid approach too.

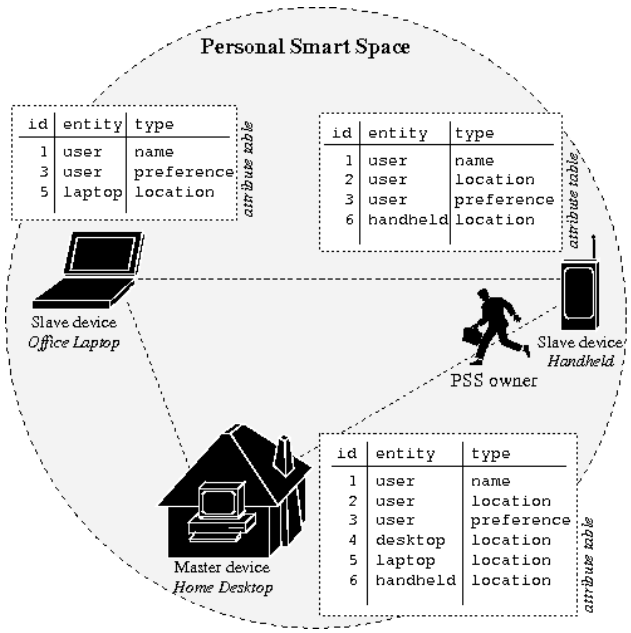

Figure 3. Example context data distribution within a PSS

Replicating this type of context information in all devices of a PSS ensures that it will be available even when the master device is currently unreachable. Furthermore, any of the aforementioned

device roles may additionally be flagged as active to indicate that this is the device the PSS owner is currently interacting with. As the active device is more likely to interact with other personal smart spaces, dynamic context information can be cached on that particular device in order to improve the responsiveness of the CM framework.

Figure 2, illustrates a rather simple example of context data distribution across the device nodes of a PSS under the hybrid distributed/centralised context management approach.

In this example, the user's desktop computer has been elected as the master device. As the user is currently away from both his home and the office, his handheld computer is considered the active device.

Thus, apart from his static context information (user preferences) and the device-specific data (handheld location), the CM of this device additionally stores dynamic context information (user location). The master device, however, aggregates context information from all device nodes within this PSS.

6 CONCLUSIONS

The scenarios discussed show how useful personalised services can be in a Personal Smart Space (PSS). The delivery of personalised services relies on the implementation of a context management subsystem so that services can utilise context data to interact with the user at the appropriate time and in the appropriate way, personalised to the user's requirements. The context management subsystem must gather and store this context data and must also maintain a context history in order to provide data to other components that provide learning and reasoning functionality.

A key research focus of the Persist project is the elimination of islands of pervasiveness where a user has no access to personalised context information while disconnected from a fixed smart space so it is necessary to store context information on each device in a user's PSS. Current device limitations prevent the replication of all context information across all devices and it was with this limitation in mind that a hybrid approach was chosen in which each device stores context information that is relevant to that device as well as static context information, while a master device stores context information from all devices in addition to context history. The complete set of context information is available on the master device to support algorithms that perform learning and reasoning for the PSS and a limited set of context is available on slave devices to allow services to access context information even when the device is separated from the rest of the PSS. When context information is not directly available on the device or in other devices in the PSS, it can be inferred.

The combination of distributed context information with personalised context aware services allows the realisation of a truly useful personal smart space that accompanies a person as they move between smart locations.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 215098 of the Persist (*PERsonal Self-Improving SmarT spaces*) Collaborative Project.

REFERENCES

[1] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, Vol. 8, No. 4, pp. 10-17 (2001).

[2] P. Remagnino, G.L. Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm", IEEE Transactions on Systems, Man and Cybernetics, Part A, Vol. 35, No. 1, pp. 1-6 (2005).

[3] R. Singh, P. Bhargava, S. Kain, "State of the art smart spaces: application models and software infrastructure", ACM Ubiquity, Vol. 7, No. 37, pp. 2-9 (2006).

[4] I. Roussaki et al., "Persist Deliverable D2.4: Initial architecture design", Technical report, PERSIST (FP7-ICT-2007-1) (2008).

[5] S. Loke, "Context-Aware Pervasive Systems: Architectures for a New Breed of Applications", Auerbach Publications, 1st edition (2006).

[6] A. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing, Vol. 5, No. 1, pp. 4-7 (2001).

[7] S. McBurney, N. Taylor, H. Williams, "Giving the User Explicit Control over Implicit Personalisation", in Procs. of Workshop on Intelligent Pervasive Environments (under AISB'09), Edinburgh, Scotland (2009).

[8] K. Frank, P. Robertson, S. McBurney, N. Kalatzis, I. Roussaki, M. Marengo, "A Hybrid Preference Learning and Context Refinement Architecture", in Procs. of Workshop on Intelligent Pervasive Environments (under AISB'09), Edinburgh, Scotland (2009).

[9] T. Buchholz, A. Kupper, and M. Schiffers, "Quality of context: What is it and why we need it", in Procs. of Workshop of the HP OpenView University Association (HPOVUA'03), Geneva, Switzerland (2003).

[10] C. Pils, I. Roussaki, T. Pfeifer, N. Liampotis, N. Kalatzis, "Federation and Sharing in the Context Marketplace", in Procs. of Workshop on Location- and Context-Awareness (LoCA'07), Lecture Notes in Computer Science, Vol. 4718, pp. 121-138, Springer Berlin / Heidelberg (2007).

[11] K. Frank, M. Rockl, P. Robertson, 'The Bayeslet Concept for Modular Context Inference', in Procs. of 2nd IEEE International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'08), pp. 96-101, Washington, DC, USA (2008).

[12] N. Kalatzis, I Roussaki, N. Liampotis, M. Strimpakou, C. Pils, "User-centric inference based on history of context data in pervasive environments", in Procs. of 3rd ACM International Workshop on Services Integration in Pervasive Environments (SIPE'08), pp. 25-30, New York, USA (2008).

Protecting the Privacy of Personal Smart Spaces

Kajetan Dolinar[1], Elizabeth Papadopoulou[2], Nicolas Liampotis[3], Yussuf Abu-Shaaban [2] and Ioanna Roussaki[3]

1 INTRODUCTION

The purpose of privacy protection is dual, as it aims to both prevent, as well as cure privacy breaches. The starting point to develop such a privacy protection framework is the investigation and the definition of the privacy breach concept. A widely acceptable answer to this is provided by the privacy protection regulation, which however does not provide a precise enough definition of the privacy breach concept. The most common principles of privacy protection can be summarised in the following list [1]:

- **Principle of fair and lawful processing** (Article 6(1), letter a): *"Any processing of personal data should be carried out in a fair and lawful way with respect to the data subjects."* (where a *data subject* is a legal or natural person to which the data refer)
- **Finality principle or Limitation principle** (Article 6(1), letter b): *"Personal data must be collected for specified, explicit and legitimate purposes and may not be further processed in a way incompatible with those purposes."*
- **Data minimisation principle or Proportionality principle** (Article 6(1), letter c): *"Processing of personal data should be limited to data that are adequate, relevant, and not excessive."*
- **Time minimization principle** (Article 6(1), letter e): *"Data should be kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the data were collected or for which they are further processed."*
- **Notification principle** (Articles 10, 11): *"Data controller or his representative has to identify himself to the data subject and notify data subject about the personal data being processed, stored or further disclosed to any third party."*
- **Principle of data subject consent** (Article 2, letters a, h; Article 7, letter a; Article 8): *"User consent is required for a legitimate data processing by any data controller. The user consent is defined as 'any freely given specific and informed indication by which the data subject signifies his agreement to personal data relating to him being processed'."*
- **Principle of right to access personal data** (Article 12): *"Data subject has right to access (and rectify) the data collected about him, to get informed about the intended processing and the logic behind the intended processing of the data."*
- **Principle of right to object processing of personal** data (Article 14): *"Data subject has right to object processing of personal data (subject to certain constraints: compare articles 7, letters e and f)."*

Another approach to this problem is to start by studying the set of known contemporary problems regarding privacy protection. The problems in privacy protection have been thoroughly analysed by various research projects. For example, the following list of privacy protection problems has been identified by [2], which is the result of a quite exhaustive study:

- *Working from home*: monitoring of employees.
- *Digital rights management*: how much it interferes with the privacy of individuals?
- *ID theft and liability*: automated payments make it easy to spend money quickly, distance contracts do not offer the same guarantees, trust and confidence as in physical commerce.
- *Data laundering*: companies are paying a lot of money for personal and group profiles, while there are market actors in position to sell them.
- *Personal profiling*: a lot of people do not realise how much personal information they are constantly giving out when engaging in online transactions.
- *Inadequate profiling*: people are victims of an inadequate profiling based on false data or processing.
- *Disproportional request for personal information*: often data controllers require a volume information disproportionally large for the purposes of business.
- *Spyware and personal preferences*: spyware is a frequent way to steal data and intrude privacy.
- *Advertising and spam*: spam not only takes time and provokes irritation, but can also influence and infringe someone's private world.

These are the real cases of privacy breaches and they cannot be mitigated unless a very sophisticated and efficient approach is taken, which should be based on state of the art technology and also consider the social models applicable. There is a sufficiently wide variety of technologies, tools and solutions to address those privacy breaches. Thus, a systemic protection should be possible.

In PERSIST we will not elaborate on each of these technologies, but will rather be interested in the overall framework of how those technologies and certain social models can fit together to prevent or cure those privacy breaches. Emphasis will be given on the abstract model and the required protocols and formal languages that need to be developed, in order to make the entire idea realizable, as well as on some specific solutions, as for example trust & reputation modelling.

[1] SETCCE (Security Technology Competence Centre), Slovenia. Email: kajetan.dolinar@setcce.si.

[2] Department of Computer Science, School of Mathematical and Computer Sciences, Heriot Watt University, UK. Email: {E.Papadopoulou, Y.Abu-Shaaban}@hw.ac.uk.

[3] School of Electrical and Computer Engineering, National Technical University of Athens, Greece. Email: { nliam, nanario}@telecom.ntua.gr.

The rest of this paper is structured as follows. Section 2 elaborates on the overall architecture of the PERSIST privacy protection system. Subsequently, Section 3 discusses the concepts of privacy preferences and privacy policies, while it describes how these are exploited in order to conduct a privacy policy negotiation process. Then, Section 4 introduces the role of identities in PERSIST and presents the proactive identity selection process. Section 5 briefly exposes the trust management framework designed in PERSIST, investigating aspects regarding the trust management requirements, the trust representation, and the trust management architecture. Finally, in Section 6, conclusions are drawn over the entire privacy protection approach designed.

2 ARCHITECTURE

There are two parts of the PERSIST privacy protection system: the part concerned mostly about protection of privacy-sensitive information before the actual disclosure (referred to as *a-priori* protection) and the part intended to provide some protection or remedies in case of privacy breaches after the data have already been disclosed (referred to as *a-posteriori* protection).

A-priori protection is based entirely on technologies and among others includes the following techniques:

- define privacy preferences and induce privacy policy defined in a formal language suitable for machine processing;
- analyse trust in peers for different situations, or find out about reputation of peers;
- between two peers and based on their privacy policies negotiate over the privacy-sensitive information to be disclosed and the appropriate privacy protection;
- define appropriate digital identity for representation according to the agreement from privacy policy negotiation; and
- make up direct data protection: configure access control lists, credentials or other attributes, archive documents to preserve integrity and time of creation, encrypt or obfuscate data, buy insurance policy for protection against privacy breaches.

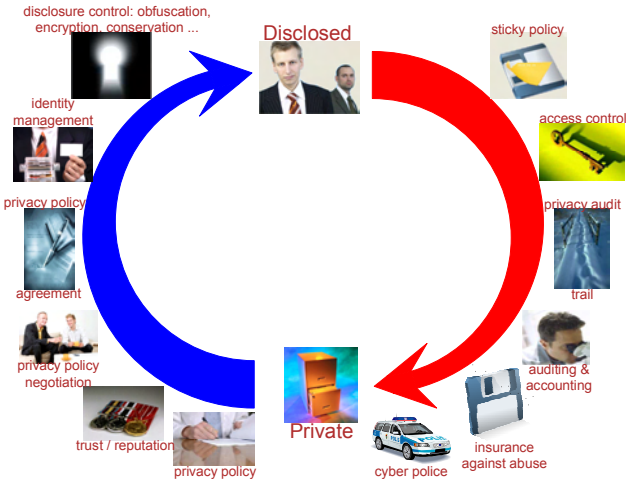

Figure 1. Privacy Protection Cycle

A-posteriori protection is a collection of techniques and social models which consist of the following:

- Attach a *sticky policy* to every data disclosed. The sticky policy is an excerpt from the agreement obtained after the privacy policy negotiation, which captures the essential privacy protection rules and parameters with respect to this particular data. The data with the sticky policy are cryptographically signed in order to preserve integrity, while data without any sticky policy are illegitimate.
- Entertain access control by credentials, and/or by access control lists, and/or by purpose (should match the purposes allowed in sticky policy).
- Use a privacy audit trail, i.e. record all actions on data (type of action, purpose, time, data and sticky policy) using a trusted hardware platform deployed on every node where processing of private identifying information goes on.
- In case of suspicion on a privacy breach, authorities can audit the privacy audit trail by investigating collective output of interesting trusted hardware platforms.
- If privacy breach has actually occurred, insurance compensates the curtailed person and penalizes the perpetrator.
- Finally, in case of severe privacy breaches, police and court take over.

This way the privacy protection is rounded into a cycle of systemic privacy protection, referred to as *privacy protection cycle*, which is illustrated in Figure 1. The *a-posteriori* part of the cycle is left mostly in a form of a concept and the majority of concrete design was made for the *a-priori* part.

We recognize several architectural components which bring about the functionalities required for a successful operation of the *a-priori* part of privacy protection cycle:

- *Policy Management*: provides interface for specifying privacy preferences, is able to produce privacy policies based on privacy preferences, and is able to negotiate agreements between two privacy policies.
- *Identity Management*: is able to produce digital identities that correspond to the extent of data and the degree of protection defined in the privacy policy negotiation agreement, it remembers the transactions where the digital identities are or were used, and it controls their lifecycle.
- *Trust Management*: uses appropriate model for collecting trust beliefs and for calculating reputation for each community member, is used in initial evaluation of peers during privacy policy negotiation, in making service recommendations, when deciding to join a group or when taking a group decision to accept a new member, etc.

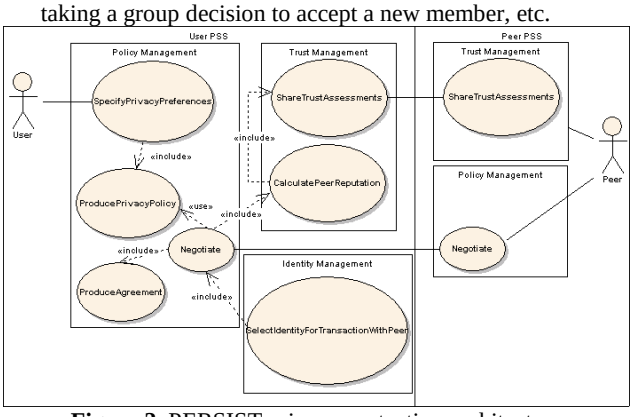

Figure 2. PERSIST privacy protection architecture

How these components interact and under which settings they are used can be seen in the diagram of Figure 2. The diagram is biased towards the User, however the situation is symmetric with respect to the Peer. It should be noticed that there are some less significant parts that are omitted in order to improve clarity. The `SelectIdentityForTransactionWithPeer` use case is invoked by a higher process of discovery of the Peer's resource of interest to the User and actuation of that resource. The `SpecifyPrivacyPreferences` use case is the only point of interaction for the User, as all the other use cases do not require the involvement of users.

3 PRIVACY POLICIES & PREFERENCES

This section describes the process of Privacy Policy Negotiation, along with the privacy policies and privacy preferences that are used to conduct the negotiation process.

3.1 Privacy Policy for Services

One general approach to protect the user privacy involves checking with a service what access to the personal user data it requires, before it is executed. Thus, when a user requests a service, the user's PSS communicates with the service and negotiates with it over the access it needs, subject to the user's requirements.

In order to accomplish this, the system uses the concept of privacy policies. These are complex data structures that define the user's requirements regarding the negotiation process. This process is referred to as *Privacy Policy Negotiation*. However, because of their complexity, they are difficult to create and maintain for the non-expert end-users. As a result, the concept of privacy preferences has been introduced. These are much simpler to understand and can be maintained via a user-friendly process. These are defined by the end-user and are used to generate the corresponding privacy policies automatically, which are far more complex.

3.2 Privacy Policy Negotiation

Privacy Policy Negotiation is the process that takes place when a user initiates the use of a service. Before the user actually starts using the service, the PSS of the user and of the service provider have to agree on the terms of usage. This requires a negotiation based on the service's privacy policies and those derived from the user's privacy preferences.

During a privacy policy negotiation, the two parties involved (the user's PSS and the Service Provider's PSS) exchange the privacy policies. Based on these, they negotiate the terms where the two policies do not match. The policies that dictate the actions to be taken during a negotiation are created on the fly after evaluating the corresponding privacy preferences. Figure 3 roughly describes the Privacy Policy Negotiation process. Assume that Abby's PSS advertises a Travel Agent Service. Jack is currently seeking to book his next holiday when his PSS receives Abby's Travel Agent Service Advertisement and requests to use it (Step 1 in the diagram). Abby's PSS responds to Jack's PSS with its Privacy Policy statement (Step 2 in the diagram). Upon receiving Abby's Travel Agent service Policy statement, Jack's Policy Negotiator creates a list of Jack's personal data, to which Abby's PSS is requesting access, and passes this list to the Policy Manager along with Abby's PSS identifier, requesting that a privacy policy is generated for this purpose. The Policy Manager retrieves all the preferences related to the attributes that Abby's PSS is requesting access to and evaluates them. The evaluation process includes requesting a trust evaluation of Abby's PSS and its Travel Agent service. The outcome of the trust evaluation is fed into the preference evaluation, where it appears as a condition. After the evaluation process is complete, the outcomes of the preferences are merged into a single privacy policy document expressed in XACML (http://xml.coverpages.org/xacml.html) and returned to the Policy Negotiator (Step 3 in the diagram).

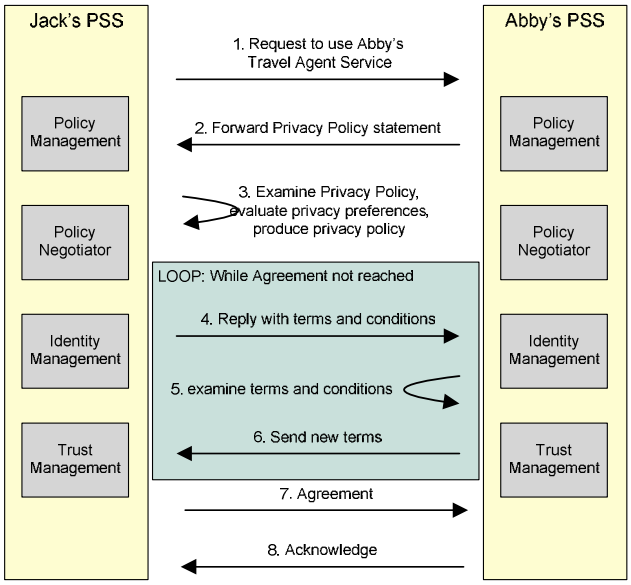

Figure 3. Privacy Policy Negotiation

Based on the customised privacy policy document, the Policy Negotiator edits Abby's Policy document under each statement indicating an agreement or a disagreement. Where a disagreement is stated, the reason is also included and terms and conditions Abby's PSS has to agree to so that access can be granted. The revised policy document is sent back to Abby's PSS, where the same process is repeated until an agreement is reached (Steps 4, 5 and 6 in the diagram). After the agreement is reached, a copy is kept by both parties for their records (Steps 7 and 8 in the diagram).

3.3 Privacy Preferences

The Privacy Preference lies in the heart of the privacy architecture of a PSS and expresses the privacy-related actions that need to be taken in certain situations. The privacy preference model consists of two main classes; the Privacy Preference Condition and the Privacy Preference Outcome. The Privacy Preference Condition describes the situation under which the Privacy Preference Outcome should be applied. Conditions are built up from elementary comparisons that check the values of attributes that describe the current context of the user at any particular point in time. Apart from the obvious and most commonly used attributes such as location, day of week and time, other parameters include features of the services that are being used at a given point in time by the user. Any parameter

that can be represented as a key-value pair can be used in a condition in the Privacy Preference Condition.

In the context of privacy, the trustworthiness of a service, with which a user is interacting, can be very important. The issue of trust in a PSS is further explored in Section 5.

There are two types of Privacy Preferences: the Privacy Policy Negotiation Preference (PPN preference) and the Identity Selection Preference (IS preference). The format of the Condition part of both types of preferences is the same. The difference between them lies in the format of the Outcome. The Outcome part of the PPN preference is used to produce the Privacy Policy that is used for conducting the PPN process described in the previous section. The Outcome of the IS preferences is used to select an identity from the pool of digital identities available to the user for a specific purpose. The IS preference is discussed in the next section. An introduction on the Identity Management (IDM) of PSSs is given in Section 4.

The PPN preference refers to one specific piece of personal information that has been requested. The Outcome of this type of preference suggests either to allow or to deny access to this piece of information. Each piece of information held in the user's PSS can have one or more Privacy Policy Preferences associated with it. Every time a piece of information is requested, the Policy Management component evaluates all the privacy policy preferences associated with it. It is possible that the outcomes of the privacy preferences conflict with each other. In this case, the system selects the "strictest" preference outcome.

4 IDENTITIES

4.1 Identity Lifecycle

Identities in PERSIST are used to group certain attributes of the user data and permit services to access them. The structure of the identity is never revealed to the services that use them. They are structures used internally in the Privacy Architecture of PSSs that maintain policies mapped to them and record historic transactions exploited for learning privacy policy preferences and identity selection preferences. The following diagram illustrates the Identity lifecycle in PERSIST.

An identity is created when the user requires a service execution, but the privacy components do not have a valid identity to map to the requirements of the services as specified in the privacy policy agreement. The Identity Management then creates an Identity to be used during the usage of this service. Once the Identity is created, the IDM attaches restrictions to the usage of this identity. The restrictions are extracted from the agreement reached at the end of the privacy policy negotiation process. At this point, an Identity reaches the "Valid" state. An Identity becomes "Active" when the service is given permission to start accessing its attributes and becomes "Inactive" when it is not used by any of the currently running services. The "Invalid" state applies to an Identity temporarily during the Identity Selection process, when the IDM tries to match an identity from the pool of identities to the requirements specified in the policy agreement and identifies as "Invalid" all identities that do not match these requirements. An Identity can take the state of "Unavailable" when certain circumstances such as context conditions render it unusable. The users themselves are also provided with the option to select a specific identity that should only used in a specific place (e.g. home or office environment) and nowhere else. Finally, an Identity is "Revoked" when the user deletes it from the system. In this case, the Identity is not deleted completely but exists in a "Revoked" state. This is due to the fact that the system needs to have a record of the attributes of that identity to map it to the historic transactions that are kept for processing.

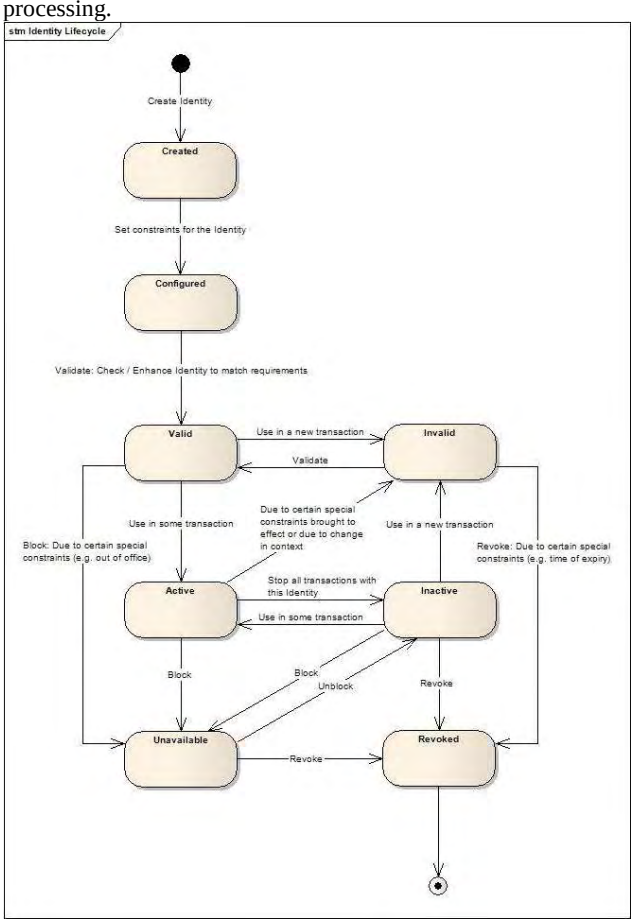

Figure 4. Identity Lifecycle in PERSIST

4.2 Identity Selection

The Identity Manager is the component that manages the Identities of the user. The management of the Identities includes maintaining both offline and run time information about the identities at all times. This includes the status of the identities as specified in the lifecycle diagram, the current transactions in which each of the identities is used, the attributes that are assigned to each identity and the restrictions that apply to each identity. It is also responsible for selecting the Identity to be used in each transaction based on the Identity Selection preferences. The Identity Selection process occurs right after the Privacy Policy Negotiation process. The Identity Selection process involves finding the list of valid identities from the pool of identities and evaluating the Identity Selection preferences to determine the exact identity to use. The process of Identity Selection is shown in Figure 5. The "Validation" process is the process that finds all the identities in the set of identities that match a set of requirements. These requirements are extracted from the Agreement that was reached during the policy negotiation. This agreement and information about the requestor

are provided to the Identity Manager by the Privacy Manager when it requests that an Identity is selected to be used in a transaction (Step 1 in the diagram). The Validation process returns a list of "Valid" identities. If the list of "Valid" identities returned is empty, the Identity Manager creates a new Identity that satisfies all the requirements specified in the agreement (Step 3b in the diagram). If one or more "Valid" identities are found, the Identity Manager requests the evaluation of Identity Selection preferences from the Policy Manager that determines which of these Identities should be used (Step 3a in the diagram). The Policy Manager evaluates the identity selection preferences that apply to all the identities in the list and determine which one is the best identity in this case (Step 4a in the diagram).

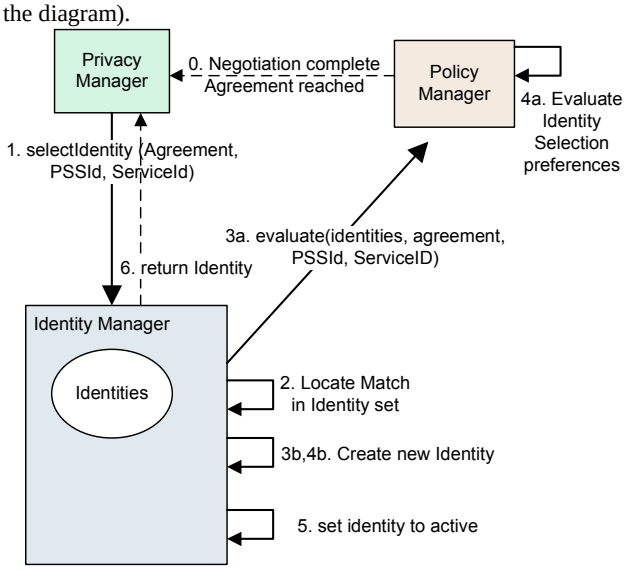

Figure 5. Identity Selection in PERSIST

It is also possible that the Policy Manager suggests that a new Identity needs to be created, due to the fact that one or more conditions do not allow any of the valid identities to be used. In this case, the Identity Manager creates a new Identity (Step 4b in the diagram). At this point, an Identity is selected to be used and the Identity Manager marks this Identity as "Active" (Step 5 in the diagram). The Identity is returned to the Privacy Manager and the service is allowed to execute on the PSS (Step 6 in the diagram).

4.3 Identity Selection Preferences

The Identity Selection Preferences are used to select the most appropriate Identity from the pool of existing Identities by evaluating a number of conditions that describe the environment of the user. As explained in section 3.3, the format of the Identity Selection preferences is almost the same as the Privacy Policy preferences. Both have the Preference Condition and the Preference Outcome parts. The Preference Condition part has exactly the same format. However, the Preference Outcome is quite different. The Outcome of an Identity Selection preference specifies in a given situation whether or not a specific identity can be used or not. For every Identity of the user, it is possible to have more than one preferences that dictates the conditions under which this identity should be used or not.

As described in section 4.2, when the IDM requests the evaluation of the Identity Selection preferences, it provides the list of "Valid" Identities in this particular instance, the requestor details and the agreement reached during the negotiation process. The Identity Selection Preference is associated with the requestor and the set of attributes the requestor needs to access. The Outcome suggests one or more Identities that would be suitable in any situation given the details of the requestor and the list of data that will be accessed.

5 TRUST MANAGEMENT

The PERSIST architecture specification was based on a scenario-driven approach, as outlined in [3]. The need for a trust management framework in this architecture became evident after analysing the identified use cases and the interaction patterns that appear in these scenarios. This section elaborates on the trust management framework that has been designed for the needs of Personal Smart Spaces.

5.1 Trust Management Requirements in PERSIST

This subsection briefly describes the requirements that should be addressed by the trust management framework in PERSIST.

Trust-based PSS negotiation: PSS negotiation is the process where two PSSs decide whether to interact with each other and is thus considered as the starting point for any collaboration between PSSs. This decision should be based on the trustworthiness of each PSS owner, i.e. the person or legal entity on whose behalf the smart space operates, as it determines the services -if any- to be made available by each party.

Trust-based privacy policy negotiation: Upon successful completion of the PSS negotiation, privacy policy negotiation takes place before allowing one party to use a service provided by the other, i.e. the visited PSS. More specifically, the privacy policy associated with the service of the visited PSS is first evaluated against the privacy preferences of the requestor and then the two parties negotiate on the personally identifying information that needs to be made available to the service in question. This process should not rely solely on the trustworthiness of the owner of the visited PSS, but the trustworthiness of the provided service itself should also be considered.

Trust-based service recommendations: The recommender system of a PSS can provide its owner with suggestions on a particular type of service based on his/her previous choices or those of other PSSs. The problem of trust is inherent in collecting and evaluating these suggestions. Therefore, the trust management framework should support the assignment of a confidence level to each service recommendation based on the trustworthiness of the party that provides the specific service.

Trust-based PSS grouping: One of the key features of PSSs is their ability to form groups for sharing context information, services and other resources. This raises substantial new privacy challenges as a member of a PSS group does not only share personal information with a service, but, at the same time, provides access to this information to other members of the group. From the point of view of a non-member, its decision whether to join a group should rely on the trustworthiness of group members, as well as, that of the services that are shared among PSS group members. On the other hand, the

trustworthiness of a PSS requesting to gain membership should also be considered prior to acceptance. It is therefore evident that the trust aspect is of key importance in PSS grouping.

Decentralised trust management: PSSs are capable of operating in both infrastructure and ad-hoc networks. This introduces additional requirements on the trust management framework of a PSS. More specifically, in order to support the ad-hoc environment, PSSs should neither rely on a centralised trust management system nor require maintaining global knowledge on the trustworthiness of every PSS. Thus, an efficient distributed trust management framework needs to be established.

Collaborative trust evaluation: The trust evaluation performed by a PSS regarding other PSSs or services should be based not only on the experience and evaluation of it own interactions, but also on those of other trustworthy PSSs. In order to support this, the trust management framework should employ mechanisms to monitor PSS interactions (direct experiences) and collect trust calculations (recommendations) from other PSSs. Effectively, trust will increase through successful interactions and degrade otherwise.

Automatic trust (re)assessment: Given the dynamic nature of trust, its assessment and reassessment should be performed in an automatic fashion allowing PSSs to constantly update their trust relationships in accordance with the behaviour of the trust objects. This would protect PSSs from parties that used to be trustworthy but have become less trustworthy or even malicious over time.

5.2 Trust representation

Trust is a very broad concept and the lack of consensus on a representation scheme is evident in the literature. In fact, the interpretation of trust has been compared by [4] to the story of the six blind men who perceived an elephant differently depending on the part of its body that they touched (e.g. as a rope because of its tail, etc.) [5]. Nevertheless, in this section, we focus on the PERSIST perspective on trust representation. The basic domains of trust relationships in personal smart spaces are defined, while the trust measurement scale is discussed. Finally, a high-level view of the trust management framework architecture is presented.

5.2.1 Trust domains

The trust management framework should allow an entity, i.e. a PSS, to express the trustworthiness of other entities within a domain. Based on the requirement analysis presented in section 5.1, five basic trust domains have been identified:

1. Trust domain D_1 expresses the trustworthiness *t(A,B)* assigned to **PSS B** by **PSS A**.
2. Trust domain D_2 expresses the trustworthiness *t(A, S)* assigned to **service S** by **PSS A**, where S is provided by **PSS B**.
3. Trust domain D_3 expresses the trustworthiness *t(A,G)* assigned to **Group PSS G** by **PSS A**.
4. Trust domain D_4 expresses the trustworthiness *t(A,P)* assigned to a **non-PSS party P** (e.g. provider, operator, user, developer, administrator) by **PSS A**.
5. Trust domain D_5 expresses the trustworthiness *t(A,R)* assigned to **resource R** (e.g. service, network, content) by **PSS A**, where R is provided by a **non-PSS party P**.

It should be pointed out that the above domains express one-way trust relationships. We also assume that trust is not symmetric, thus, *t(A,B)* is in principle not equal to *t(B,A)*. Moreover, supporting trust domains on a service level, allows for a more fine-grained level of PSS trustworthiness. So, for example, the weather forecast service may be assigned with a different trust value than the restaurant recommendation service that are both provided by the same PSS.

5.2.2 Trust scale

Trust, in its simplest form, can be binary, i.e. a given entity may be either fully trusted or not trusted at all [6]. However, most trust models use a single value over a specific range in order to represent trust. In fact, a variety of different trust value ranges has been proposed. In [7], the trust value is taken from [0, 1], while in [8] a scale with discrete trust levels in the form {−n, ..., −1, 0,+1, ...,+n} is considered.

In such trust representation schemes, the highest value usually denotes full trust, while the lowest one stands for full distrust. The problem in this approach is the representation of distrust as a result of negative experiences, as opposed to distrust based on lack of previous experience or knowledge of it.

The above issue can be addressed by maintaining two distinct order structures on trust values, i.e. a trust ordering and an information ordering, as proposed by [9]. The former represents the degree of trustworthiness, as previously described, while the latter expresses the degree of precision of trust information. More specifically, the first (lowest) element in the information ordering represents no knowledge and the last (greatest) element represents certainty. A similar solution is proposed by [10]. In their work, trust is represented as a 3-dimensional metric that includes values for belief (trust), disbelief (distrust), as well as, uncertainty. The problem of differentiating between distrust and simple uncertainty is further aggravated in ad-hoc environments where interacting with entities not known beforehand is very common. Thus, the uncertainty factor is also considered in the trust representation scheme of the PERSIST trust framework.

Furthermore, another aspect that is considered by this framework is the freshness/outdatedness of collected data that contribute to the trust evaluation. Thus, the trust model designed is capable of distinguishing between the same level of trust, evaluated based on the same type and number of interactions, when these interactions have taken place in different points in time (e.g. the last month or one year ago). This feature addressed the last of the presented requirements, thus protecting PSSs from parties that used to be trustworthy but have become less trustworthy over time or vice versa.

5.3 Trust Management Architecture

The components of the Trust Management framework are illustrated in Figure 6. The Trust Manager component controls various operations in the framework and provides an interface to other PERSIST components, such as the Policy Management and the Recommender System. Direct trust information is maintained in the Trust Repository. The automatic (re)assessment of direct trust between pairs of PSSs (or between a PSS and an external actor in general) is handled by the Direct Trust Evaluator, which is uses the Risk Engine and the Learning Management for this purpose. Inferring trust when there is no direct trust relation between PSSs is supported by the Trust Inference component. Inferring the trust relationship between a group of PSSs and an

individual PSS is handled by the Group Trust Evaluator component.

The rest of this section is structured as follows. Initially, a description of the issues that should be considered during the process of direct trust assessment is provided, which is followed by a brief overview of the issues regarding the inference of trust in PERSIST.

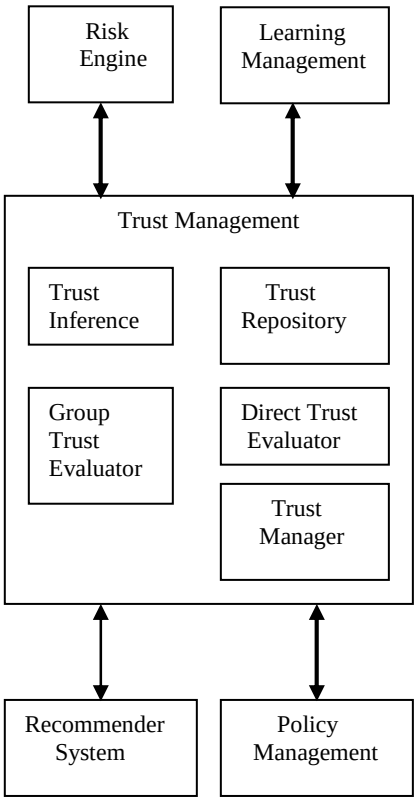

Figure 6. Trust Management Architecture

5.3.1 Direct trust assessment

A trust relation between two PSSs is initially created after their first interaction. As described in Section 5.1, the automatic reassessment of trust is a requirement of the Trust Management framework. When a PSS interacts with an entity, a number of factors could influence the (re)evaluation of the PSS's direct trust in that entity after the interaction has been completed, such as:

- Factors that could be learnt from the history of interaction including the number of previous interactions between the PSS and the entity, the frequency of interaction and interaction duration.
- Another factor that could influence the (re)evaluation of trust after interaction completion is the interaction risk. A high risk interaction with an entity that was completed successfully should lead to a high trust value allocated to that entity. Similarly, the trust in an entity should be significantly reduced if a high risk interaction fails.
- Trust ageing is another factor that could be considered when assessing direct trust between PERSIST entities. A trust value should be decreased if a long time has passed since the last interaction (the last trust update).

5.3.2 Trust inference

The use of trust inference is required to estimate PSS A's trust in PSS B, for a specific domain, when there is no direct trust relation from A to B in this domain. A number of trust inference algorithms have been devised based on aggregating the trust values along the trust path(s) between A and B. The algorithm proposed by [11] includes a set of trust propagation schemes based on the eigenvalue propagation and weighted linear combination. The EigenTrust algorithm presented in [12] calculates the reputation of peers in a peer-to-peer file sharing network. Each peer reputation is given by the local trust values assigned directly to it, weighted by the reputation of the assigning peers. An approach is proposed in [13] to determine trust by randomly selecting a path in a decision tree that is constructed based on trust information.

A number of issues could be tackled in devising an algorithm for the trust inference problem in PERSIST. This includes:

- **Trust grouping**: As described in section 5.1, grouping of PSSs is supported in PERSIST. For a PSS to join/interact with a PSS group, the trust relation should be inferred. Trust inference becomes a one-to-many problem where the collective trust of the PSS group members needs to be estimated.
- **Aggregating trust across domains**: A wide range of services with different domains could be supported in the PERSIST platform. Thus, a large set of domains will be associated with the trust relations between PSSs. The trust inference algorithm could consider aggregating trust values from related domains.

6 CONCLUSIONS

This paper elaborated on aspects of the privacy management framework designed by the PERSIST project Consortium to address the requirements of Personal Self Improving Smart Spaces. More specifically, the concept and the various types of privacy breach in computing/communication environments have been explored before the brief presentation of the overall architecture of the PERSIST privacy protection system. Additionally, the privacy policy negotiation process has been exposed, as well as the role of privacy preferences, privacy policies, and identities. Finally, the trust management framework designed is presented, where the trust management requirements, the trust representation, and the trust management architecture have been briefly described.

The design of the privacy protection framework in PERSIST has been finalised in March 2009. Up to this point, detailed interface specification has been produced, while all necessary UML diagrams have been generated (i.e. class, component, sequence and state diagrams). According to the PERSIST workplan, the prototype implementation of the privacy protection framework will be complete by May 2010. Thorough testing and evaluation will follow, in order to validate this proof-of-concept implementation. Of course, there are various other aspects that need to be addressed before this privacy protection framework can be widely used, such as protection from spyware, pushed advertising and spam, theft of identity, data laundering and abuse of personal information provided by individuals during online transactions. However, it is a firm belief of the authors that the research work described in this paper will make

a small step towards the introduction of privacy-aware personal smart spaces in the wide market.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 215098 of the Persist (*PERsonal Self-Improving SmarT spaces*) Collaborative Project.

REFERENCES

[1] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 24 October 1995 on the protection of individuals with regards to the processing of personal data and on the free movement of such data, Official Journal L No. 281, (1995).

[2] Y. Punie, S. Delaitre, I. Maghiros and D. Wright, (eds.) "SWAMI Deliverable D2: Dark scenarios in ambient intelligence - Highlighting risks and vulnerabilities", Technical report, PERSIST consortium, (2005).

[3] K. Frank et al., "PERSIST Deliverable D2.1: Scenario Description and Requirements Specification", Technical report, PERSIST consortium, (2008).

[4] R.J. Lewicki and B.B. Bunker, "Trust in relationships: A Model of trust development and decline", in *Conflict, Cooperation and Justice*, pp. 133–173, Jossey-Bass Publishers, (1995).

[5] Wikipedia. Blind men and an elephant. http://en.wikipedia.org/wiki/Blind_men_and_an_elephant, (2008).

[6] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system", in *Procs of the 10th ACM International Conference on Information and Knowledge Management* (CIKM'01), pp. 310–317, New York, USA, (2001).

[7] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web", in *Procs of the 2nd International Semantic Web Conference*, pp. 351–368, (2003).

[8] H. Prade, "A Qualitative Bipolar Argumentative View of Trust", in *Procs of the 1st International Conference on Scalable Uncertainty Management* (SUM'07), pp. 268–276, Berlin, Heidelberg, (2007).

[9] V. Cahill et al., "Using Trust for Secure Collaboration in Uncertain Environments", *IEEE Pervasive Computing*, Vol. 2, No. 3, pp. 52–61, (2003).

[10] X. Li, M.R. Lyu, and J. Liu, "A Trust Model Based Routing Protocol for Secure Ad Hoc Networks", in *Procs of IEEE Aerospace Conference*, pp. 1286–1295, Big Sky, Montana, USA, (2004).

[11] R. Guha and R. Kumar, "Propagation of Trust and Distrust", in *Procs of the Inter national World Wide Web Conference*, New York, USA, pp. 403–412, (2004).

[12] S. Kamvar, M. Schlosser and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks", in *Procs of the ACM World Wide Web Conference*, Budapest, Hungary, pp. 640–651, (2003).

[13] U. Roth and V. Fusenig, "How Certain is Recommended Trust-Information", in *Procs of the ACM World Wide Web conference*, Edinburgh, UK, (2006).