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.

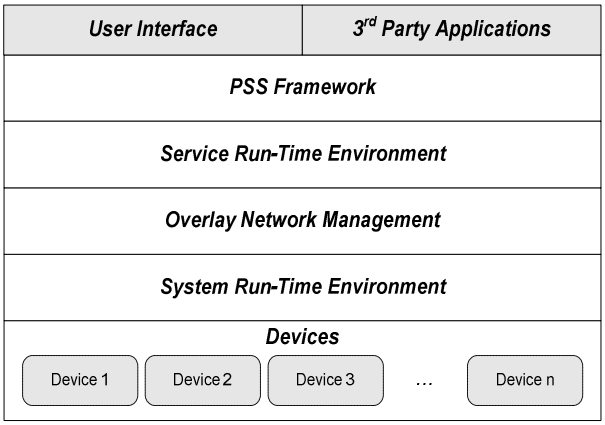

Figure 1. The high level architecture of Personal Smart Spaces

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.

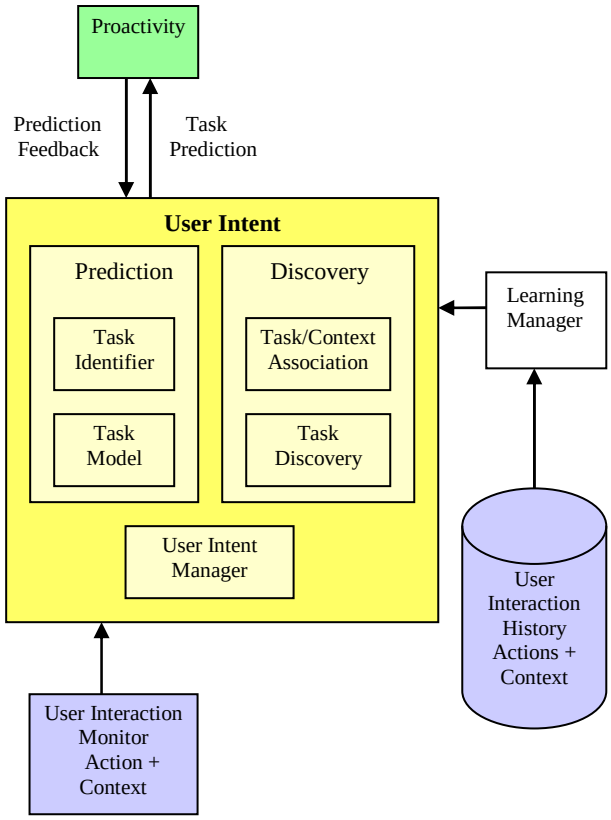

Figure 2 The User Intent System Architecture

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.

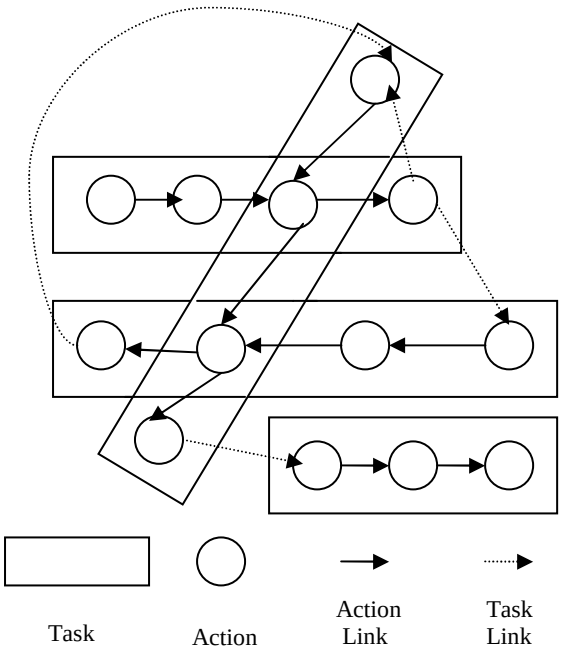

Figure 3 Task Model Discovery

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.

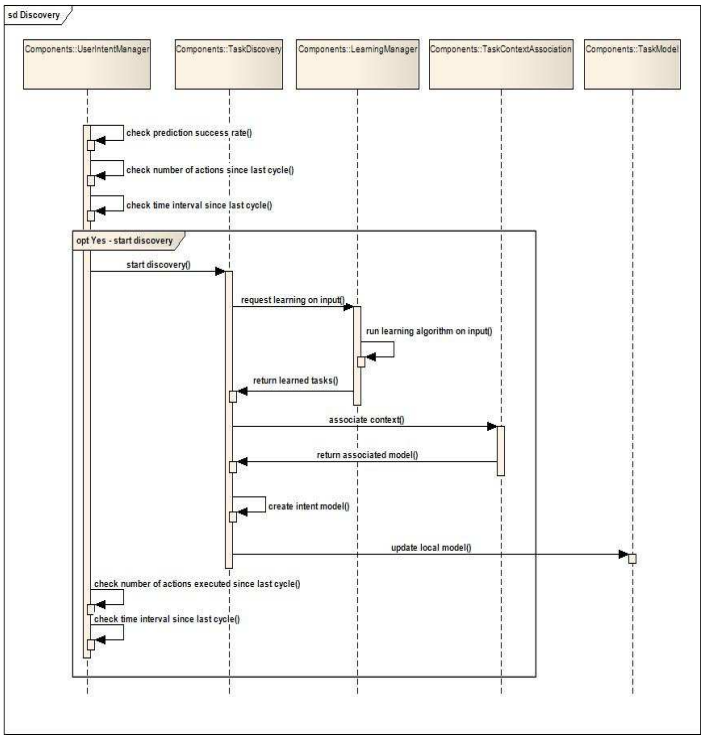

Figure 4 The Task Discovery Cycle

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

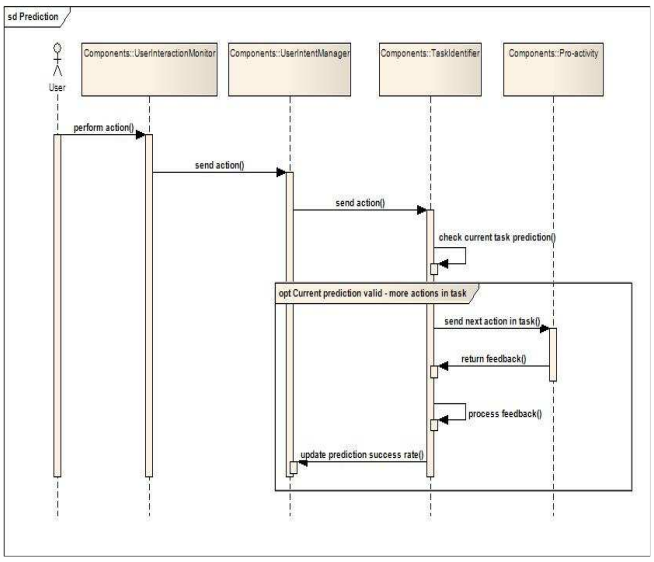

Figure 4 Prediction When Current Predicted Task is Valid with More Actions in Task

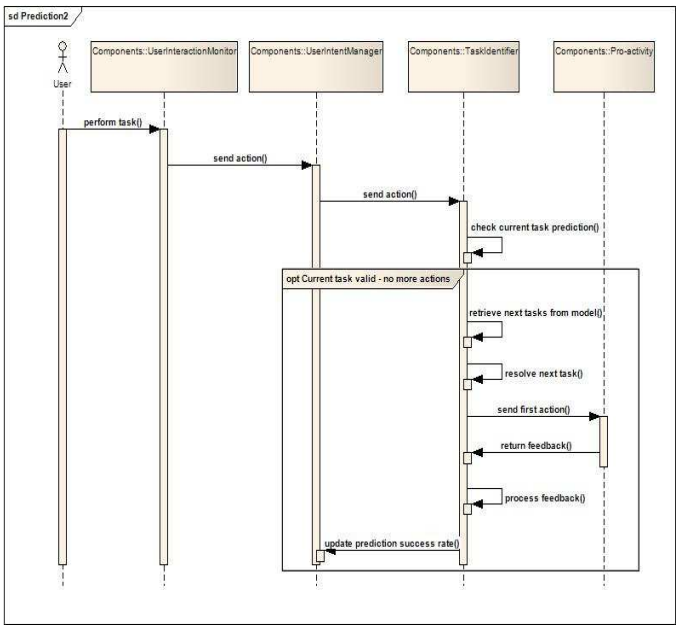

Figure 5 Prediction When Current Predicted Task is Valid with No More Actions in Task

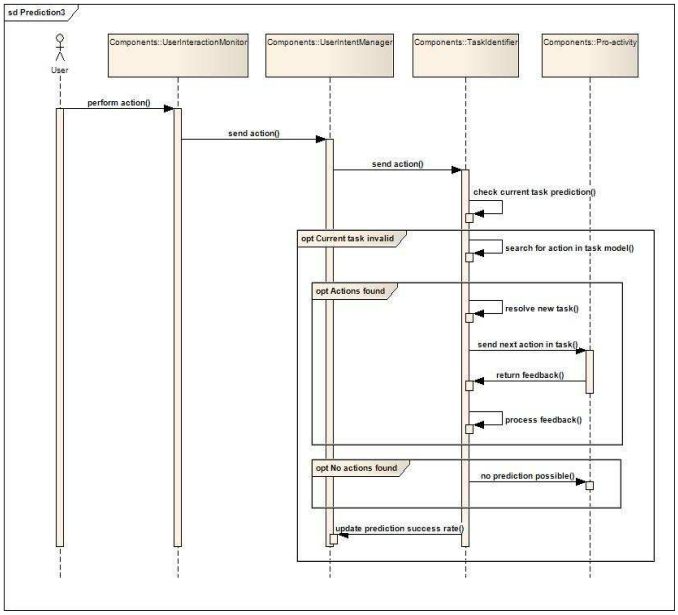

Figure 6 Prediction When Current Predicted Task is Invalid

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.