In the IEEE International Conference on Pervasive Services, The First IEEE International Symposium on Pervasive Computing and Applications（SPCA06), August 3-5, 2006, Urumchi, Xinjiang, P.R. China

A Pervasive Computing Programming Approach for Non-Technical Users

Jeannette S Chin[1], Vic Callaghan[1], Graham Clarke[1]
[1]*Department of Computer Science, University of Essex, United Kingdom*
jschin@essex.ac.uk; vic@essex.ac.uk; graham@essex.ac.uk

Abstract

Our homes are rapidly being filled with diverse types of products ranging from simple lighting systems to sophisticated entertainment systems, all adding to the functionality and convenience available to the home user. In this paper we introduce a variant of end-user programming in the form of a toolcalled PiP that empowers non technical end users to be able to "program" the functionality of their personal pervasive computing environments to suit their particular needs. The paper also includes a user evaluation that shows end-users find this approach to be a useable and enjoyable experience.

Keywords: Pervasive Computing, End-user Programming, Programming-by-Example, Show-Me-by-Demonstration, Deconstruction, Digital Home, Intelligent Environment, Extreme Programming, End-User Empowerment.

1. Introduction

Our homes are rapidly being filled with diverse types of products ranging from simple lighting systems to sophisticated entertainment systems, all adding to the functionality and convenience available to the home user. However, as impressive as current gadgets and appliances may seem, the home user is set to experience nothing short of a revolution in the nature and capability of their home environment brought about by adding embedded computers and network connections to both existing electronic artefacts and those hitherto passive. This development opens up the possibility for hundreds (or more) of communicating home devices, cooperating in communities serving the occupant; so called "Pervasive Computing".

In this paper we introduce a variant of an end-user programming tool primarily targeted at home based pervasive computing environments, which aims at empowering the non technical end users to be able to "program" the environment to suit their particular needs. We employ "*show-me-by-example*" techniques allowing the users to accomplish such tasks not just without the need for them to write any code but also to show the system how they would like their environment to behave by simply demonstrating some examples. We call this methodology Pervasive interactive Programming (PiP).

2. Related Work

The area - Pervasive Computing has been receiving a great deal of attention recently, although most of the research and tools available currently are aimed at developers rather than end users. For example, some research work has focused on building infrastructure or framework to support the dynamic changes nature of the environment [3, 4, 5, 9, 11, 12] whilst other work has targeted building high-level abstractions to support tasks [6, 7, 8, 10] or providing user-friendly tools to simplify a person's use of the system [2, 13, 14].

The most relevant research to PiP is now described: Humble [15] uses a jigsaw, metaphor, enabling users to "snap" together puzzle-like graphical representations as a way of building applications; Truong's CAMP project [16] places the end-users at the centre of the design experience by using a fridge magnet metaphor, together with a pseudo-natural language interface, that collectively enables end-users to realize context-aware pervasive applications in their homes; Media Cubes [1] which offers a tangible interface for programming an environment where each face of a cube is represented by a set of program structures. "Programming operations" are achieved by turning the appropriate face of the cube towards the target device; The Alfred project [17] utilises a macro programming approach to enable a user to compose a program via "teaching-by-example" using verbal or physical interactions. Whilst these are very useful approaches, they are either not flexible enough to support the end users' intuitive physical interactions or places a high cognitive load upon the users eg. utilising methods such as macros requires users to adhere to a strict ordering of instructions or otherwise the system will fail.

3. Motivations

The motivation behind PiP was to create a system that maximized user's control and operational transparency (engendering a sense of trust) and enabled them to "*program"* their own environment, without any detailed technical knowledge (thereby empowering user creativity). This motivation was driven by experience with autonomous agent based systems where users' desired for controls were partly taken, fears on personal privacy, information gathering and its usage were expressed [18].

To date most of the research directed at this area has focused on streamlining the use of the input languages or metaphor-based GUI interfaces, aiming at simplifying the use of the applications for the end users. Currently most end-user programming tools for ubiquitous environments are still based on the procedural programming metaphor and require the user to mentally manipulate constructs that would be familiar to most programmers thereby placing a significant cognitive load on the user. As we have been inspired by the ease that people perform daily routine tasks (eg. switch on the light when the room gets dark, muting the TV sound when the telephone rings etc), we decided to direct our approach at finding a way of programming that was natural and mimicked familiar everyday practices as much as possible.

4. Pervasive interactive Programming (PiP)

PiP is primarily aimed at *end users* in any *service-rich* pervasive environment. We assume that services are offered from networked devices supported by underlying protocol layers which are not described in this paper. PiP provides a platform that utilises the physical user environment as the programming environment thereby enabling the user to "program" the functionality that they require to suit their particular needs. Thus, with a minimum effort, the end user, who has no technical expertise, is able to produces customised effects on groups of pervasive devices in the environment that can usually only be achieved by conventional programming.

4.1. PiP Concepts

Definition: the term "device" used in this section refers

Figure 1 -A pervasive computing environment

to any application that runs on the network which is able to either initiate or react to commands relating to a service (physical or information) it offers, which typically resides in appliances, embedded-processors or PCs.

4.1.1. Pervasive Device and Applications

(Figure 1) is a technology-rich environment heavily populated with network aware devices and services. It is centred around the concept of services that provide functions to accomplish particular tasks. The success of these tasks is partly attributed to the ability of a device to communicate their internal states. With a supporting software framework, these services are discoverable, and therefore accessible to the environment in which they reside. Generally devices in a pervasive world would offer at least one service but there is no restriction on the number of services a device can offer. An example of a supporting framework is Universal Plug and Play (UPnP)[1].

4.1.2. A Deconstructed Model – Virtual Device

As devices and their services in pervasive environments are discoverable and accessible, a number of possibilities emerge. For instance by aggregating sets of services it becomes possible to form "virtual devices". This new model of "virtual devices" offers to radically change the conventional perception of a "device" as the functional units that make up current devices are shared. The rationale is that a "virtual device" made up of the functionalities of other devices could accomplish some tasks, that individual device was not capable of. "Virtual device" could have an impact on how developers produce their products. More importantly, end users could leverage this "device and service rich" pervasive world to create their own "virtual devices" to suit their needs. We refer to such communities or "virtual devices" as **M**eta **Ap**pliance**s/Ap**pliaction**s (MAps)** and the approach as the deconstructed appliance model.

4.1.3. MetaApplicances (MAp)

The concept of a MAp is a core concept in PiP. From a logical perspective, a MAp has primitive properties and a collection of *Rules* that determine the behaviour of the coordinating devices and, as a consequence, the environment, which is the end user's personal space. Rules are essentially a marriage of 2 different types of actions, namely 'Antecedent' (condition) and 'Consequent' (action). Each action (whether it is an 'Antecedent' or a 'Consequent') has the property of a "virtual device". The 'Antecedent' of a Rule can be described as "if" while the 'Consequent' of a Rule can be described as "then". A Rule can contain 0-n 'Antecedents' and 1-n 'Consequents', and a MAp legally can contain 0-n Rules (as Rules can be added later by the end user).

MAps are a non-terminating process and require no specific user expertise for their formation. They are created *under the directions of end-users* to provide the sort of behaviour functionalities that they like. They can

[1] UPnP network technology allows personal computer and consumer electronics devices to advertise and offer their services to network clients. More details UPnP forum at: http://www.upnp.org/

Figure 1. A Typical Pervasive Environment

be represented graphically and be visible to the user who created them, either at the time of creation or later when they can be retrieved, shared, executed, or removed on demand. Until a MAp is terminated, it will retain the functionalities that the user originally created (ie. it is a continually running process).

4.2. PiP System Architecture

PiP is designed to work in real time within a pervasive environment (Figure 2). The communication between PiP, the end user and the environment is via an eventing mechanism, thus PiP has an event-based object-oriented asynchronous architecture. Unlike macro languages, that are commonly used in desktop computing end-user programming paradigm, where sequence of actions is significant, PiP assumes the logical sequence of actions is not important.

Figure 2. PiP high level architecture

PiP leverages UPnP™ technology as its middleware and communication protocol, enabling simple and robust connectivity among devices and PCs. It has modular framework comprising six core modules, which work together to support real-time network computation (see figure 3),

The core modules are:

a) "Interaction Execution Engine" (IEE) – this module has a network control point and is responsible for device discovery, service events subscription, and performing network action requests.

b) "Eventing Handler" (EH) – this module acts like a middle-man, responsible for interpreting low-level network events (eg device discovery), device service events (eg service state changes) and high-level events that generate from "PiPView" caused by the user interactions. Its main role is to communicate events between interested modules.

c) "Knowledge Engine" (KE) - this module is responsible for assembling and instantiating a "virtual device" before storing them in the Knowledge Bank. It is also responsible for updating the device's current status, as well as maintaining an up-to-date version of the Knowledge Bank.

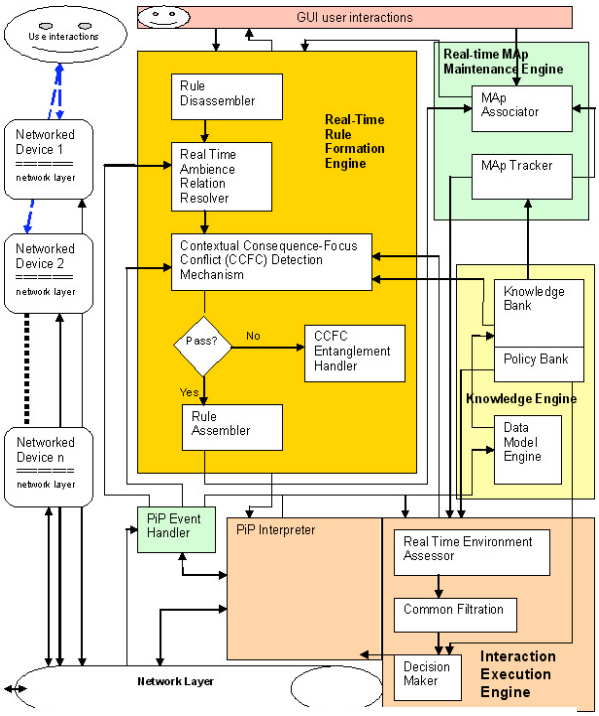

Figure 3 -PiP Modular Architecture

d) "Real-time MAp Maintenance Engine" (RTMM) — is a process that maintains the records of current and previously created MAps.

e) "Real-Time Rule Formation Engine" (RTRF) – this module is responsible for assembling rules based on user interactions within the "demonstration" mode[2].

f) "GUI" – A graphical interface called "PiPView" that the user can use to make inspections of the environment, compose/delete Maps/Rules etc, interact with the system and control physical environment.

5. Show Me by Example

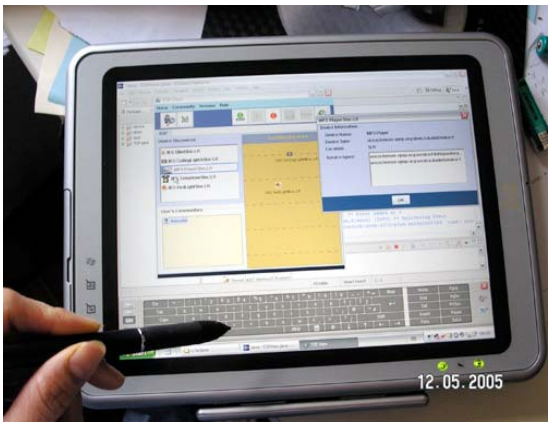

Figure 4. PiP on tablet view

This section illustrates how the assembled system actually works. In PiP, not just "PiPView" (the system

[2] A "demonstration" mode begins when the user clicks "ShowMe" button and end when user clicks "Done!" button.

Figure 4. The iDorm2 test bed

GUI) can be regarded as a user interface, but so can other networked devices, since users interact with them during the demonstration process. Examples are: networked dimmer lights, networked telephone, network entertaining systems, network fridge etc (Figure 1). Using these devices as user interfaces, users can interact intuitively and naturally with the environment and the metaphor for programming their environment is thus very simple. The user creates a MAp (ie. the "program" that captures the functionalities of the environment) by showing the system the functionalities that the MAp should have via simple familiar interaction e.g by using a wall switch to turn on a light etc., and PiP will do the rest for the user.

Figure 5. A user demonstrates her actions via physically interacts with the devices

A MAp is created by the user "dragging & dropping" device representations through PiPView. A MAp can be given collective functionality by the user demonstrating the required behaviour by engaging in physical activities using the *real* devices, previously selected via PiPView. In PiP, the user can choose when to inform the system they are ready to begin to show (program) the devices functionalities by using any of the three methods: (1) physically interacting with the devices themselves, (2) using a UI control panels (3) a combination of the above two the choice being left to the user. Based on the actions of the user, the pervasive devices generate appropriate events and pass them to the network and PiP encodes this information as a set of rules with two parts; an antecedent (conditions) and consequent (resulting action) as it "listens" and "captures" the user's action as demonstrated.

In PiP the user is given as much freedom as possible, allowing antecedents and consequents to be formed in any number and order (ie. the user is not required to follow a rigid logical sequence of order). To execute a MAp, the user needs only to drag the MAp graphical representation and drop it into a "play" button located at the top of the PiPView. To terminate a MAp the user simply clicks on the "stop" button.

6. End Users Evaluation

An end-user evaluation was carried out in the iDorm2 (Figure 4) at the University of Essex[3], a two-bedroom apartment built to be an experimental pervasive computing environment. Five sets of pervasive devices were created for the evaluations- (1) a bed light, (2) a desk-light, (3) telephone, (4) a sofa and (5) an MP3 player. While the telephone and the sofa have embedded sensors, both the two lights have on/off switches as their interfaces. These devices were connected to a snap [19] board. The MP3 player was implemented as a software emulation run on PC with a mouse interface for manual control. All devices were run on UPnP middleware network

6.1. Evaluation Design and Procedure

The end user evaluation was designed to (1) see if users were able use PiP in a creative manner to construct MAps of their own design and (2) gather an insight into their post-trial views of PiP based on six factors: "Conceptual Understanding", "User Control", "Cognitive Loading", "Information Presentation", "Affective Experience" and "Future Potential". Our aim for the evaluations were to setup an open ended trial giving the end users as much freedom as possible to maximise the potential for creativity and see how the participants preferred to use the system. This freedom included time, methods, and tasks..

Eighteen participants (10 females and 8 males) sampled from a diverse set of backgrounds (eg housewives, students, secretaries, teachers etc) participated in the evaluation. All participants had some minimal computing experience ie. they knew how to use a mouse. Whilst 21.3% of the participants had a very good knowledge of programming, 57.4% of them had none at all.

During the evaluations, PiP was set-up to run on a winXP tablet PC (HP) that connected to the iDorm2 network via a Linksys 802.11g WIFI access point. Each trial was preceded by a 20-minutes training session to allow participant to familiarise themselves with the system. The task for the evaluation was that the participant should use PiP to program the pervasive environment to behave in the way they wanted. No specific type of behaviour for the environment was set for the evaluation, rather the participants were free to

[3] idorm2 at http://iieg.essex.ac.uk/idorm2/index.htm

create one (or more) of their own. Participants were encouraged to test out their newly created environment instantly after creation. No time limit was set either and assistance was provided where needed.

Following completion of the evaluation, a questionnaire with a scale of responses ranging from "Strongly Agree" through to "Strongly Disagree" was administrated to measure the participants' subjective judgements of PiP. Participants rated a total of seventeen statements covering six usability dimensions[4]. Data was analysed using SPSS[4].

7. Performance and Results

As our evaluations objectives were neither aimed at measuring the system nor the participants' performance in terms of time, thus there was no formal timing measurments during the trial periods. However, we observed that after a brief training session, 83% of participants were able to use PiP to "*program*" their environment (ie. creating MAps) with little or no assistance (although time taken to accomplish these tasks varied from participant to participant).

Among the means used for demonstrating user's examples, a 11% of the participants chose to create their programs using wholly GUI controls whilst 72% of them did so via physical interaction with the environment, the rest using a combination of both.

Although PiP is immune to logical order when composing MAps, 33% of the participants (mostly those with programming experience) found it convenient to use logical sequence for composing MAps whilst the remander focused on the functionalities rather than logical sequence. .

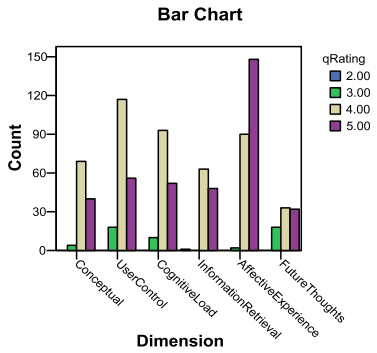

Figure 6. Dimension vs qRating crosstabulation

Various tests (using SPSS software package) were carried out to analyse the questionnaire ratings. From the results we observed that the "Affective Experience" dimension received the highest ratings (148 out of the total number of 240 cases or 61.7% received a top rating). It was discovered that the "Information Retrieval" dimension (ie information presentation) had the lowest rating ("2" was recorded) whereas all other dimensions, a "3" was the lowest recorded (Figure 6).

Table 1 shows an overall rating scale for six dimensions evaluated. For the rating range from 1 – 5, all six usability dimensions have a mean rating above 4.1, suggesting, in general, the participants found PiP useful, and their experience of *programming* the environment simple and enjoyable.

In addition we observed that 83.4% of all participants found PiP intuitive to use and 94.4% of all participants stated they felt it a rewarding experience.

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Conceptual	113	4.3186	.53894	.05070	4.2181	4.4190	3.00	5.00
UserControl	191	4.1990	.59134	.04279	4.1146	4.2834	3.00	5.00
CognitiveLoad	155	4.2710	.57332	.04605	4.1800	4.3619	3.00	5.00
InformationRetrieval	112	4.4107	.54613	.05160	4.3085	4.5130	2.00	5.00
AffectiveExperience	240	4.6083	.50596	.03266	4.5440	4.6727	3.00	5.00
FutureThoughts	83	4.1687	.76221	.08366	4.0022	4.3351	3.00	5.00
Total	894	4.3602	.59489	.01990	4.3211	4.3992	2.00	5.00

Table 1. One-Way ANOVA test on dimension vs qRating

8. Conclusion and Future Work

This paper has described our research into end-user programming for pervasive computing and the development of a programming paradigm called Pervasive Interactive Programming (PiP). which enables non-technical end-users to program the environment functionality they require within a pervasive computing environment. We have implemented a small proof-of-concept version of PiP which we evaluated on 18 users. Whilst acknowledging that the participants are only a small sample of the population, the initial results are encouraging as they show that PiP served different types of user well, allowing them to program the environment to suit their needs. Thus, we contend that this approach is usable by non-technical end-users to create their own functionalities in the technology-rich pervasive environments, such as digital homes. The current version of PiP is not a commercial version and so, for our future work, we hope to refine the system further. Also, For MAps to be portable across environments it is essential that there is a generic way of describing the capabilities of collectives of devices and services and for this we plan to refine our dComp ontology [20]..

Acknowledgement

We are pleased to acknowledge financial support from the UK DTI Next Wave Technologies and Markets programme and the University of Essex. We also wish to record our thanks to our colleagues Martin Colley, Hani Hagras and Malcolm Lear for their strong support.

[4] SPSS at http://www.spss.com/

References

[1] Hague, R., et al: Towards Pervasive End-user Programming. In: Adjunct Proceedings of UbiComp 2003 (2003) 169-170

[2] Humble, J. et al "Playing with the Bits", User-Configuration of Ubiquitous Domestic Environments, Proceedings of UbiComp 2003, Springer-Verlag, Berlin Heidelberg New York (2003), pp 256-263

[3] Tandler, P.: Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices. In: Proceedings of Ubicomp 2001: Ubiquitous Computing. Springer-Verlag, Berlin Heidelberg New York (2001) 96-115.

[4] Grimm, R. et al "Programming for Pervasive Computing Environment", Proceedings of 18th ACM, Symposium on Operating System Principles, Canada, Oct., 2001

[5] Becker, C. et al " BASE: A Micro-broker-based Middleware for Pervasive Computing", Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom03), Fort Worth, USA 2003.

[6] Wang Z, Garlan D. "Task-Driven Computing" Technical Report, CMU-CS-00-154, Computer Science, Carnegie Mellon Univ, May 2000.

[7] Masuoka, R. et al : Semantic Web and Ubiquitous Computing - Task Computing as an Example -AIS SIGSEMIS Bulletin 1(3) October 2004.

[8] Masuoka R et al "Task Computing - the Semantic Web meets Pervasive Computing," 2nd Int'l Semantic Web Conf (ISWC2003), 20-23 Oct 2003, Florida, USA

[9] Kameas, A. et al "An Architecture that Treats Everyday Objects as Communicating Tangible Components", Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom03), Fort Worth, USA 2003.

[10] Shahi, A. et al , Introducing Personal Operating Spaces for Ubiquitous Computing Environments. Pervasive Mobile Interaction Devices 2005 (PERMID 2005), hosted by 3rd International Conference on Pervasive Computing, Munich 8-13, May, 2005.

[11] Garlan, D. et al " Project Aura: Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing Magazine, April-June 2002.

[12] The PHEN project: http://iieg.essex.ac.uk/phen

[13] Drossos N. et al , **"***A Conceptual Model and the Supporting Middleware for Composing Ubiquitous Computing Applications***",** The IEE International Workshop on Intelligent Environments, University of Essex, Colchester, UK, 28-29 June 2005

[14] Tsukada, K. and Yasumura, M.: Ubi-Finger: Gesture Input Device for Mobile Use, Proceedings of APCHI 2002, Vol. 1, pp.388-400

[15] Humble, J. et al "Playing with the Bits", User-Configuration of Ubiquitous Domestic Environments, Proceedings of UbiComp 2003, Springer-Verlag, Berlin Heidelberg New York (2003), pp 256-263

[16] Truong, KN.et al " CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home", Proceedings of Ubicomp 2004, pp 143-160.

[17] Gajos K., Fox H., Shrobe H., "*End User Empowerment in Human Centred Pervasive Computing*", Pervasive 2002, Zurich, Switzerland, 2002.

[18] Chin J et al "Pervasive Information Systems: Issues for the Individual and Society", UN Second World Urban International Conference on "*The Role of Cities in an Information Age*", September 13-17, 2004, in Barcelona, Spain.

[19] SNAP http://snap.imsys.se/

[20]. J, Chin et al "*Virtual Appliances for Pervasive Computing: A Deconstructionist, Ontology based, programming-By-Example Approach*", The IEE IE05, Colchester, UK, 28-29 June 2005.