Presented at Intelligent Environments 2008, Seattle, 21-22 July 2008

A PROGRAMMING-BY-EXAMPLE APPROACH TO CUSTOMISING DIGITAL HOMES

J.S.Y Chin, V.Callaghan, G.Clarke
University of Essex, UK, jschin@essex.ac.uk

Keywords: End-user programming, Programming-by-Example, Deconstruction, Digital Homes, Ubiquitous Computing.

Abstract

The arrival of the digital home, signals a new era for both manufacturers and consumers. It raises the possibility that people may be able to create their own unique digital home functionalities, by combining elemental network services to create sophisticated "virtual appliances" that satisfy their individual needs and desires. We present taxonomy of approaches for customising digital homes and use this to introduce a novel computational methodology, that we term 'Pervasive Interactive Programming' (PiP), that allows non-technical end-users to build their own "virtual appliances" for customising their 'electronic environments' without the need to write program code. We show this approach is novel in that it firstly allows the user to translate mental concepts into functions (which differs to agent based approaches using historical behaviour data to form functions) and secondly it moves "programming-by-example" from single processor computers into real-time distributed embedded computing environments. Finally we report on a small evaluation which showed it was possible for non-programmers to use these tools to customise the electronic functionality of their digital homes.

1 Introduction

The arrival of the digital home, awash with networked services and appliances, signals a new era for both manufacturers and consumers. People need no longer be dependent on manufacturers to supply a limited selection of monolithic fixed functionality appliances rather, in addition to buying off-the-shelf appliances and applications, they will be able to create their own unique virtualised appliances and applications (meta –appliances –applications called MAps), by combining elemental network services to deliver functionality that satisfy their individual needs and desires. This new deconstructionist model of appliances will present manufacturers and consumers with new marketing, research and lifestyle opportunities. This user driven customisation model leads to challenges such as how non-technical people can translate their needs into meta functionalities (coordinated service actions) and how to support important, human qualities such as creativity. Various user requirement studies have found that there is a need for harmonious cooperation between appliances and the ability for the user to customise their home [20][21][22]. Even for the same individual, their desire may change over time and, sometimes, in a seemingly non-deterministic way. In a digital home functionality not only arises from individual appliances but from the coordination of multiple network-appliance actions making a more holistic, environment-wide, functionality. It is this coordinated activity that gives each environment a unique behaviour; a behaviour that can be configured to meet the occupants unique needs. Thus, a step towards achieving this vision is addressing the challenge of how to empower non-technical end-users to customise such coordinated activities in digital home environments; a key challenge that this paper addresses. Our solution is based on an approach derived from the "programming-by-example" in which all the end-user needs to do, is simply to *show* the system the behaviour required from their 'virtual appliance' via physical and graphical interaction with the environment.

1.1 Customising Digital Homes

The home of the future might contain tens or even hundreds of network based services, some provided by physical appliances within the home, others by external service providers. These services could be used in a variety of ways, for example manufacturers might run remote diagnostics as part of a more efficient appliance service provision, or they could gather usage data that would help them improve the design of their products. More radically, from a customer's perspective, customers might be able to *explicitly* customise the functionality of digital homes that arises from coordinating sets of networked services. Such personalised functionalities could follow a person as they move across environments (with certain functions existing as default functions in all homes, such as telephones heating etc). Indeed such customised functions, in the form of 'soft-objects' (see section 3.1.3) could even be traded. This paper concerns the development of intuitive methods that non-technical end-users can use to customise the functionality of their Digital Homes. In particular this paper explores the hypothesis that ideas similar to programming-by-example (see next section), which was primarily targeted single non-networked computer environments via a graphical user interface, can be adapted and built upon to enable people living in digital home to customise the coordinated functionality of sets of distributed service providing computer based appliances. This involves addressing a wide range of issues ranging from the underlying computer science to human factors.

1.2 Motivation

A number of significant studies have investigated digital home requirements. In one study the Samsung

Corporation, in cooperation with the American Institutes for Research, conducted a study aimed at identifying smart home requirements by interviewing and monitoring people in South Korea and the USA [21]. Their findings included the need for harmonious cooperation between appliances and ease of use. In relation to the work presented in this paper one particularly important requirement they discovered was the need for people to be able to customize their home. The same need was uncovered from a 3 year study on Finnish people undertaken by Tampere University Hypermedia laboratory [22].

The goal of this work is to address this need for people to be able to customise the coordinated functionality of appliances and services in their digital homes. In addition our motivation was driven also by experience with autonomous agent based systems where users' desire for control, fears on personal privacy, information gathering were expressed [18]. Thus the aim was to create a system that maximized user's control and operational transparency (engendering a sense of trust) and enabled people to customise the functionality of their digital-homes, without any detailed technical knowledge (thereby empowering user creativity). We have done this by exploring how an end-user programming paradigm, programming-by-example, originally developed for "desktop computer environment" can be applied to distributed computing environments that underpin digital homes. We call this approach Pervasive interactive Programming[1] (PiP); and the principles are presented in this paper.

2.0 A Taxonomy of Approaches for Customising Digital Homes

Researchers have investigated a number of different approaches to customising digital homes ranging from agent through to user driven methods. One of the most widely used methods of endowing coordinated behaviour to network appliances is via rules [23] [24]. Rules are fundamental to determining how a given appliance or service interacts with other appliances or services [23]. A fundamental distinction between different approaches is how these rules are formed. We suggest that rules in digital home systems take the following forms; *hard-coded rules* (usually created by the developers or manufacturer), *machine generated rules* (created from intelligent agents, artificial intelligence or machine learning mechanisms) and *user generated rules* (created from end-user programming). Other researchers have arrived at similar classifications albeit using different headings and rationales [25]. Thus, we have adopted a '*rule formation'* based taxonomy as a means of grouping the different approaches to approaches to customising digital homes.

[1] UK Patent No: GB 0523246.7

2.1 Hard-coded rule approaches

In this approach rules (program code) are fixed in the appliances before they are supplied to users (usually by the developer or manufacturer). Thus, the appliances or services, whilst being controllable by people using the system, and sometimes offering many automated features (eg switching lights off when there is nobody in a room) do not allow people to alter the controlling programs (especially relating to creating or altering conditional events). Examples of this approach include context aware computing, such as Microsoft's EasyLiving project which, enables systems to adapt by switching between pre-programmed routines or states [26].

2.1 Machine generated rule approaches

These approaches are based on the use of intelligent agents, artificial intelligence or machine learning mechanisms. In general these are pre-emptive decision making systems that use models derived from past behaviour to 'guess' the users need. The automated rule generation is considered by many as a considerable advantage, while others contend that there can never be a perfect match between past actions and future needs (as users will sometimes want to alter their behaviour) and that this gap will remain a source of frustration where systems need to be overridden sometimes by users. There are many excellent examples of this approach including The MavHome [27], The iSpace [24] and the Neural Home [28].

2.3 User generated rule approaches

This approach commonly referred to as *end-user programming* is characterised by the use of techniques that allow non-technical people to create "programs" [29]. This captures the essential concept underpinning the vision to enable people to customise the functionality of their own digital homes.

One approach to end-user programming is to transpose conventional programming constructs and algorithms into some form of iconic objects (eg graphical icon, physical artefact) which people 'assemble' to create applications without being confronted with code. Another way to reach this goal is using the so-called *Programming-by-Example*, approach, introduced by Smith in the mid-seventies, where desired computational behaviour is demonstrated via concrete examples by the end-users, rather than in the form of abstractions (eg programming code) [30].

Originally programming-by-example was aimed solely at single desktop environments but recently a number of researchers have started to explore how these ideas might be applied to digital homes, made up of distributed embedded computers (usually integrated into household appliances). For instance, Humble [2] uses a jigsaw, metaphor, enabling users to "snap" together puzzle-like graphical representations as a way of building applications. The HYP system [31] enables users to create

applications for context-aware homes using a mobile phone based graphical interface. Media Cubes [1] offers a tangible interface for programming an environment in which each face of a cube is represented by a set of program structures ("programming operations" being achieved by turning the appropriate face of the cube towards the target device). Truong's CAMP project [16] placed 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 enabled non-technical people to realize context-aware ubiquitous applications in their homes. Perhaps, the closest work to that presented in this paper is the Alfred project, developed by MIT, which employed the concept of "goals", and "plans" to allow users to compose a program via "teaching-by-example". The system was intended to utilise a macro programming approach which could be created by the user via verbal or physical interaction. However, according to the developer of the system, Gajos, no formal studies were completed and the work appears to have been cut short when he moved from MIT to the University of Washington. [17].Whilst macros are simple and intuitive, their dependence on strict order can make them less fragile and susceptible to failure, especially in applications where order of events is irregular or unpredictable which is one area that PiP, described in the following text, differs.

3.0 Pervasive interactive Programming

Pervasive interactive Programming **(**PiP) is primarily aimed at non-technical people living in a digital home. We assume that services are offered from networked devices supported by underlying protocol layers which are not described in this paper (eg UPnP). PiP provides a platform that utilises the physical user environment as the programming environment thereby enabling the people to customise the functionality of their digital home to suit their particular needs. Thus, with a minimum effort, an end user (resident of a digital home), who has no technical expertise, is able to produce customised effects on groups of coordinating network devices in a digital-home that could previously only be achieved by conventional computer science programming.

3.1 PiP Concepts

Figure 1 depicts a technology-rich environment heavily populated with network aware devices and services. It is centred on 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. An example of a supporting framework is Universal Plug and Play (UPnP)[2].

[2] 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 - PiP high level architecture

3.1.2 Virtual & Deconstructed Appliances

Home appliances are made up from numerous services (eg a TV is composed of a video display, audio transducer, media gateway, control interface etc). Thus, when appliances are connected to a network it becomes possible to "deconstruct the appliance" and offer these basic services to other network users, who in-turn, may combines them with other networked appliance services in numerous different ways. For instance, by aggregating sets of services it becomes possible to form new composite services or, as we chose to describe them, "*virtual appliances*". This new model of "virtual appliances" offers to radically change the conventional perception of an "appliance" and even has the potential to disrupt the current appliance market. The rationale is that a "virtual appliance" made up of the functionalities of other devices could accomplish some tasks, that individual device was not capable of. "Virtual appliances" 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 appliances" to suit their needs. This concept goes beyond home appliances and includes any network service. Thus, we prefer to refer to such communities or "virtual appliances" by the more generic name of **M**eta **Ap**pliance**s/Ap**pliactio**ns (MAps)** and the conceptual approach as the '*deconstructed appliance model*'.

3.1.3. Meta -Appliances -Applications (MAps)

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 MAp. 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 non-terminating processes and require no specific user expertise for their formation. They are created *under the directions of end-users* to provide the functionalities they desire. They can 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 behaviour that the user originally created (ie. it is a continually running process). MAps can be viewed as "soft-objects" that define the membership and behaviour of a collection of services that constitute a virtual appliance. As such MAps can be designed, owned, copied, carried or traded.

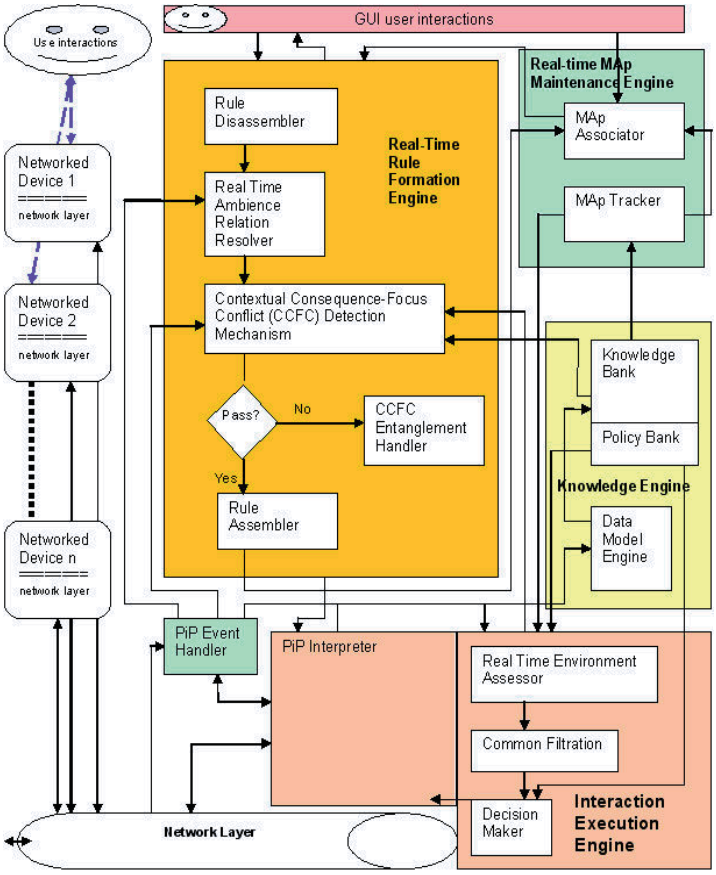

Figure 2 -PiP Modular Architecture

3.2 PiP System Architecture

PiP is designed to work in real-time with a variety of networked devices within a digital-home. 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 of no importance.

PiP leverages UPnP™ technology as its middleware and communication protocol, enabling simple and robust connectivity among devices and PCs (but could, potentially, work with any middleware). The computational architecture is shown in figure 2. As can be seen from this diagram PiP has modular framework comprising six core modules, which work together to support real-time network computation.

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.

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[3].

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.

3.3 The dComp Ontology

To enable PiP to utilise devices it is necessary to provide descriptions of their capabilities; an ontology provides such a description. PiP leverages ontology semantics as the core vocabulary for its information space, generating ontology-based rule sets when a user demonstrates her/his desired tasks to the system.

Current middleware, such as UPnP falls short of providing sufficient information on the capabilities of network devices [5]. As a result ontology is being looked at as a possible way of providing sufficient information on network devices to reason about how they might be synergistically combined. The SOUPA ontology from Ubicomp [32] is aimed at pervasive computing and OWL-S, previously called DAML-S [33] is based around the notion of services but primarily targets the World Wide Web, enabling agents to evoke services thereby facilitating the automation of web tasks. Both lack support for crucial PiP mechanisms such as community, decomposed functions and coordinating actions which are essential to produce higher level meta functionality. In addition, the current SOUPA standard has only limited support for the UPnP standard which our research testbed, the iSpace, depends on.

dComp Community Class
SoloCommunity
NotJointCommunity
PersistentCommunity
TransitoryCommunity
CommunityDevice
Rule
FixedRule
PersistentRule
NonPersistentRule
Preceding
Device

dComp Rule Class
Rule
FixedRule
PersistentRule
NonPersistentRule
Preceding
Device

Table 1 – Example classes in dComp ontology

Thus, we have had to devise our own ontolological representation, dComp (deconstructionist & community programming). However, wherever possible we have sought to adopt suitable ontology for our other needs. For example our Person, Policy and Time ontology are adopted from Ubicomp SOUPA ontology and we have implemented this using the OWL language which is widely used (especially for the semantic Web). Finally, in dComp, preferences are referred as 'situated preferences', which is akin to Vastenburg's 'situated profile' concept where he uses situation as a framework for user profile so that the values of the profile are relative to situations [34]. In the current implementation we have defined a few classes to support the notion of community and rules (Chin 2005) By way of an example, Table 1 shows the dComp community and rules classes; the full specification is available online [3]. Figure 3 illustrates how dComp can be applied to define virtual appliances.

```
<com:TransitoryCommunity rdf:ID="JCTV">
 <com:communityID>Tran-JCTV</com:communityID>
 <com:communityName>JC TV</com:communityName>
 <com:communityDescription>The first JC testing
TV</com:communityDescription>
 <com:timeStamp rdf:datatype="&xsd;dateTime">2004-09-
06T19:43:08+01:00</com:timeStamp>
 <com:hasOwner>
  <person:Person>
   <person:firstName rdf:datatype="&xsd;String">Jeannette</person:firstName>
   <person:nickname rdf:datatype="&xsd;String">JC</person:nickname>
   <person:gender rdf:resource="#Female"/>
  </person:Person>
 </com:hasOwner>
<com:hasCommunityDevice>
 <com:CommunityDevice>
  <device:deviceUUID>UUID:PHLCRT17</device:deviceUUID>
 </com:CommunityDevice>
 <com:CommunityDevice>
  <device:deviceUUID>UUID:PHLAudioMMS223</device:deviceUUID>
 </com:CommunityDevice>
 <com:CommunityDevice>
  <device:deviceUUID>UUID:NetGem442</device:deviceUUID>
 </com:CommunityDevice>
</com:hasCommunityDevice>
</com:TransitoryCommunity>
```

Figure 3 A dComp definition of a TV virtual appliance

4 Show Me by Example

This section illustrates how the assembled system actually works. In PiP, in addition to the graphical interface "PiPView", the networked appliances form the user interface, since people interact with them during the demonstration process. Using these devices as user interfaces, users can interact intuitively and naturally with the digital-home and the metaphor for programming their environment is thus very simple. The user creates a MAp (ie. the description that captures the functionalities of the coordinating appliances) by demonstrating the functionalities that the MAp should have via simple familiar interaction (e.g by using a wall switch to turn on a light etc).

In our prototype a MAp can be created by physical demonstration or via the graphical interface (see figure 4). Using the graphical interface the user "drags & drops" device representations through PiPView. A MAp can be given collective functionality by the user demonstrating the required behaviour by engaging in graphical activities manipulating icons or 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 (customise) 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.

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

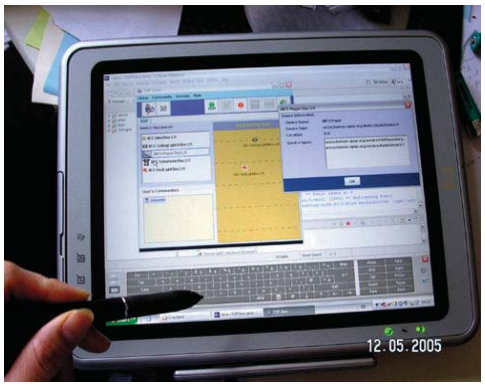

Figure 4 -. The graphical version of PiP

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.

5 End Users Evaluation

As many researchers have pointed out, the real world is a very different place than the lab [4].Thus the end-user evaluation was carried out in the iSpace (Figure 5) at the University of Essex [4], a two-bedroom apartment which is the closest to a natural home setting that was available to the project. In this 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 board [19]. The MP3 player was implemented as a software emulation running on PC with a mouse interface for manual control. All devices were operated on UPnPmiddleware network.

Figure 5. The iDorm2 test bed

5.1 Evaluation Design and Procedure

The end user evaluation was designed to:

(1) determine if users were able use PiP in a creative manner to construct MAps of their own design and

(2) gain an insight into the participants 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 was to setup an open ended trial giving the end users as much freedom as possible to maximise the potential for creativity (one of the virtues of PiP) 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. Figure 6 shows one of the evaluation participants using PiP to customise her own environment functionality. All participants had some minimal computing experience (ie. they knew how to use simple office applications). 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 windows XP tablet PC that connected to the iSpace 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 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[5].

6 Performance and Results

As our evaluations objectives were neither aimed at measuring the system nor the participants' performance in terms of time, no formal timing measurements were made during the trial periods.

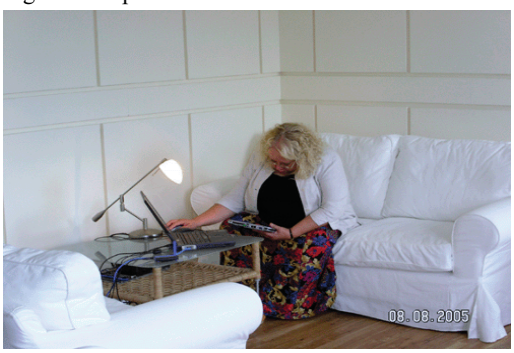

Figure 6. A user teaches the system via physically interacting with the devices

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

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

However, we observed that after a 20 minute 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).

	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

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.

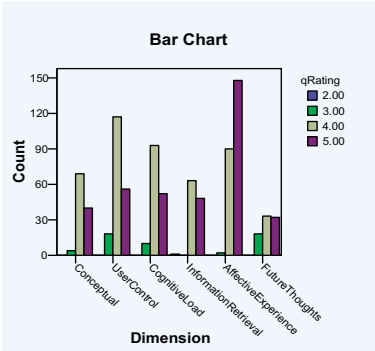

Figure 6. Dimension vs qRating cross tabulation

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 remainder focused on the functionalities rather than logical sequence.

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). The findings of this evaluation are summarised in table 1, providing an overall rating for six dimensions evaluated. The rating range is 1 – 5, with 5 being the most positive rating. From this it can be seen that the "Information Retrieval" dimension (ie information presentation) scored the lowest rating whereas enjoyment of the experience had the highest rating. 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. The low "Information Retrieval" rating may be related to the design of the graphical user interface on the prototype and suggests that this is an aspect that would need to be improved for a commercial version (if a GUI was retained). The fact users enjoyed using the system was an especially pleasing finding.

7 Conclusion and Future Work

This paper has described our research into end-user programming for pervasive computing and the development of a customisation paradigm called Pervasive Interactive Programming (PiP). which enables non-technical end-users to personalise the environment functionality within a pervasive computing environment.

As part of this work we have presented a taxonomy for describing approaches to customising the functionality of digital homes based on '*rule formation';* PiP and other end-user programming methodologies belonging to the '*user generated rules'* category. In general, end user programming approaches provide a means for non-technical people to express their creative ideas, directly from their mind to the system functionality.

We have also presented a new model for providing appliances in digital homes; the so-called deconstructed appliance or virtual appliance model. This approach has the potential to disrupt the traditional market by allowing non-technical people to design their own "virtual appliances" in the form of MAps, a type of soft-object which could even be traded.

We have implemented a small proof-of-concept version of PiP which we evaluated on 18 users and all six usability dimensions returned a mean rating above 4.1, suggesting, in general, the participants found PiP useful, and their experience of *programming* the environment simple and enjoyable. Given the prototype nature of the interface, which would be vastly improved for a commercial implementation, this was a pleasing result. Thus, we contend that this approach is usable by non-technical end-users to create their own functionalities in digital homes.

Acknowledgement

This work was funded, in part, by the UK Department of Trade and Industry and the University of Essex.

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] DCOMP, "*Deconstruction and Community Based Ontology For Pervasive Computing"* (http://iieg.essex.ac.uk/dcomp/ont/dev/2004/05/ accessed 2nd Feb 2008)

[4] Johnson B, "*Do Digital Homes Dream of Electric Families: Consumer Experience Architecture as a*

Framework for Design", In book "Advanced Intelligent Environments", Springer, 2007

[5] Horan B. The Use of Capability Descriptions in a Wireless Transducer Network, *Sun Microsystems Research Labs, Report Number: TR-2005-131*, Feb 1, 2005 (available at http://research.sun.com/techrep/2005/abstract-131.html, accessed 2nd Feb 2008)

[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] Callaghan,V., Clarke,G.S., Chen, Y., *'Some Socio-Technical Aspects Of Intelligent Buildings and Pervasive Computing Research*', Intelligent Buildings International Journal, Earthscan, 1:1 (2008).

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

[20] Carsten Röcker, Maddy D. Janse, Nathalie Portolan and Norbert Streitz, "*User Requirements for Intelligent Home Environments: A Scenario-Driven Approach and Empirical Cross-Cultural Study*", *Joint sOc-EUSAI conference*, 2004, 111-116

[21] Kook Hyun Chung, Kyoung Soon Oh, Cheong Hyun Lee, Jae Hyun Park, Sunae Kim, Soon Hee Kim, Beth Loring, Chris Hass, "*A User-Centric Approach to Designing Home Network Devices*", *CHI '03 extended abstracts on Human factors in computing systems*, 2003, 648 – 649

[22] Mäyrä F, Soronen A, Vanhala J, Mikkonen J, Zakrzewski M, Koskinen I, Kuusela K, "*Probing a Proactive Home: Challenges in Researching and Designing Everyday Smart Environments*", *Human Technology Journal,* Volume 2 (2), October 2006, 158-186

[23] Zamudio V and Callaghan V, "*Facilitating the Ambient Intelligent Vision: A Theorem, Representation and Solution for Instability in Rule-Based Multi-Agent Systems*", International Transactions on Systems Science and Applications, Volume 4 ▪ Number 2 ▪ July 2008

[24] Callaghan V, Clark G, Colley M, Hagras H Chin JSY, Doctor F "*Intelligent Inhabited Environments*", BT Technology Journal , Vol.22, No.3 . Klywer Academic Publishers, Dordrecht, Netherlands, July 2004

[25] Kainulainen L "*Reasoning in The Smart Home*", AIOME, 2006

[26] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern and Steven A.Shafer, "*EasyLiving: Technologies for Intelligent Environments*", Proc of the 2nd international symposium on Handheld and Ubiquitous Computing, 2000, 12 – 29

[27] D.J. Cook, M. Huber, K. Gopalratnam and M. Youngblood, "*Learning to Control a Smart Home Environment",* Innovative Applications of Artificial Intelligence, 2003

[28] Mozer, M. C. "*The neural network house: An environment that adapts to its inhabitants*" In M. Coen (Ed.), Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments (pp. 110-114). Menlo, Park, CA: AAAI Press, 1998

[29] Cypher ***A,*** Halbert *DC,* Kurlander *D,* Lieberman *H*, Maulsby *D*, Myers *BA*, and Turransky *A, "Watch What I Do: Programming by Demonstration*" The MIT Press, Cambridge, Massachusetts, London, England 1993

[30] Smith, D. C., "*Pygmalion: A Computer Program to Model and Stimulate Creative Thought*", Basel, Stuttgart, Birkhauser Verlag. 1977

[31] Barkhuus, L., Vallgårda, A: "*Smart Home in Your Pocket*", Adjunct Proceedings of UbiComp 2003 (2003) 165-166

[32] Chen *H,;* Finin T,; Joshi*l A.* "*SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*", Int'l Conference on Mobile & Ubiquitous Systems: Networking & Services (MobiQuitous 2004), Boston, Massachusetts, USA, August 22-26, 2004

[33] OWL-S. "*Ontology Web Language for services"*, (http://www.w3.org/Submission/2004/07/accessed 7th Feb 2008)

[34] Vastenburg M, "*SitMod: a tool for modelling & communicating situations*", 2nd International Conf, Pervasive 04, Vienna Austria, April 21-23, 2004, ISBN: 3-540-21835-1