Presented at the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06), Wuhan and Three Gorges, China, September 3-6, 2006

An End User Tool for Customising Personal Spaces in Ubiquitous Environments

Jeannette S Chin, Vic Callaghan, Graham Clarke
Inhabited Intelligent Environments Group, University of Essex, UK
Email: jschin@essex.ac.uk;vic@essex.ac.uk; graham@essex.ac.uk
URL: http://iieg.essex.ac.uk

Abstract—
End-User programming is characterised by the use of techniques that allow end-users of an application program to create "programs" without any technical expertise. This paper presents a variant of end-user programming tools primarily targeting ubiquitous computing environments that allows non-technical end-users to create "programs" for customising their personal spaces. End-users do not need to write program code, or follow a rigid sequential list of actions in order to achieve results. All the end-user needs to do, is to show the system the required personal space behaviour by demonstrating it via physical interactions with the environment. The paper includes a user evaluation that shows end-users find this approach to be a useable and enjoyable experience.

Keywords: Ubiquitous Computing, End-user programming, Programming-by-Example, Show-Me-by-Demonstration, Deconstruction, Pervasive Computing, Intelligent Environment, End User Empowerment

1.0 INTRODUCTION

End-User programming is characterised by the use of techniques that allow end users of an application program to create "programs" without any technical expertise [Cypher et al]. One way to achieve this is to create new types of "scripting language" by abstracting conventional algorithms of functionalities into some form of representations (eg graphical objects) and then operating on these representations to create a program. Another way to reach this goal is using Programming-by-Example, an approach introduced by Smith in the mid-seventies, where functions for the representations were demonstrated via concrete examples by the end-users, rather than in the form of abstractions [Smith77]. Since then, end user programming has developed into a diverse set of applications [Myers90, Guibert03] most of which were aimed solely at single desktop environments, employing macro languages to help the end users to perform some tasks on the computer.

This paper presents a variant of end-user programming tools primarily targeting ubiquitous computing environments. It employs a "show-me-by-example" approach allowing non-technical end-users to create "programs" for customising their personal space in an environment. The end-users are neither required to write program code, nor follow a rigid sequential list of actions in order to 'program' their personal space. All the end-user needs to do, is simply to *show* the system their required personal space behaviour by demonstrating via physical interactions within the environment.

The goal of this work is to merge the traditional end-user programming "desktop computer environment" with the physical environment, involving multiple distributed computers, which exist in a ubiquitous environment. Using a *show-me-by-example* approach, end-users are able to create "programs" that encapsulate the actions that they perform. These "programs" can be retrieved later to recreate the behaviour of their personal space in the ubiquitous environment - on demand, or be terminated on request. We called this approach Pervasive interactive Programming (PiP).

2.0 UBIQUITOUS ENVIRONMENTS – AN AREA SUITABLE FOR END USER PROGRAMMING

With the advance of the Ubiquitous Computing [Weiser], our environments are being populated with an increasing number of networked devices and smart sensors, offering a variety of services to the people and other devices that make up these environments. The availability of network delivered services opens up the possibility of assembling composites of coordinating services thereby enabling the environment as a whole to take on a collective behaviour, creating so-called smart or intelligent ubiquitous environments.

In the home of the future we imagine they might contain tens or even hundreds of networked devices offering services. Some of these devices might take the form of physical appliances whilst others might be virtualised appliances made up of aggregated services from numerous separate devices. Remote access that networks could for example, allow manufacturers to support customers in new ways, or to gather usage data that would help them improve the design of their products. From a customer's perspective the environment might function in a smarter way, as appliances and services could coordinate actions in ways designed by manufacturers or even the customers (end-users) themselves

Figure 1. A typical Ubiquitous environment

Technical infrastructures are vital for supporting the dynamic nature of ubiquitous environments. Research in this area has been directed at physical development [Greenberg01, Ballagas03], through system architecture framework [Grimm et al, Stevenson et al], to middleware [Tandler01, Grimm et al, Becker et al, Kameas et al, Limb et-al, Garlan et al, Andersen et al]. In addition, much attention has been directed to the end-users' perspective and needs such as research aimed at helping end-users control the environment [Nichols01, Tsukada02, Humble03, Drossos05], building high-level abstractions to support tasks [Wang, Garlan02] and applying end-user task-based paradigms [Shahi05, Masuoka03, Masuoka04]. Automating the environment to match an individuals' needs, without the need for them to be explicitly involved or have any cognitive involvement has been addressed by a number of groups such as the University of Essex [Callaghan05], the University of Colorado [Mozer98] and the University of Texas [Cook04]. More recently, the theme, 'empowering end-users' has emerged with much attention focused on developing end-user programming tools [Truong et al, Gajos02], allowing the end users to program their environment.

The overall objective of ubiquitous computing is to provide an *experience* for the end users, and there is clear evidence that people have started to pay attention to the issue of user empowerment whereby people are given the capability to make programs of their own without the need of any technical expertise. The issue of agent versus human control goes beyond simple technical matters and touches on areas of ethics and human values such as who, what, when (the so-called "3 W's") should monitor, program and control our environments [Chin05]. Systems based on autonomous agents are sometimes criticised as being too much like "Big Brother" in nature. The invisible sensing, learning and communication resulting in a sense of unease, which is particularly evident in the private spaces of our homes. In this paper we take the view that the system operation should be as transparent as possible to the user, with them in firm control of the "three W's".

3.0 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 customise their own environment, without the need for detailed technical knowledge (thereby empowering user creativity)..

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, aimed at simplifying the use of applications for the 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 (albeit in a graphical or macro form) 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. In the following sections we describe one such method that is based on these principles.

4.0 Pervasive interactive programming (PiP)

PiP is primarily aimed at *end users* in *service-rich* ubiquitous environments. We assume that the services are offered from networked devices supported by underlying protocol layers, discovery/registration process etc all of which are not described in this paper. PiP provides a platform that utilises the physical user space as the programming space enabling the user to customise the functionality of their personal space (programming) by using their space. The term "program" in our approach refers to a representation of a collection of operations in the environment. It differs to the conventional computer science meaning of a "program" which can be regarded as assembling numerous lines of sequenced instructions. Our meaning of a "program" is closer to that of a non-terminating process which may also be graphically represented. However, whilst such a "program" is constructed by an end user, who has no technical expertise, it produces an effect (ie. the operations in the user's personal space) that is normally achieved by conventional programming.

4.1 BACKGROUND CONCEPTS

Definition: the term "device" used in this section refers to any application that runs on the network which is able to either initiate or react to commands relating to a service it offers (physical or information), which typically resides in appliances, embedded-processors or PCs

4.1.1 UBIQUITOUS DEVICES AND APPLICATIONS

As mentioned earlier ubiquitous environment is a technology-rich environment heavily populated with network aware devices and services (offered by the devices/applications). It is centred around the concept of services which contain functionalities that can help to accomplish particular tasks. The success of these tasks is partly attributed to the capability of a device exposing its internal states. Since devices in this environment are interconnected, services in this context are therefore tied to the physical environment itself. With a supporting software framework, these services are discoverable, and therefore accessible to the environment in which they reside. Generally devices in a ubiquitous world would offer one service but there is no restriction on the number of services a device can offer. An example of a ubiquitous device is an UPnP[1] device. Figure 2 shows a description of an UPnP light device.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:schemas-upnp-
org:device.xlighting.1</deviceType>
    <friendlyName> UPnP Light Device</friendlyName>
    <manufacturer>IIEG</manufacturer>
    <manufacturerURL>http://iieg.essex.ac.uk/</manufacturerURL>
    <modelDescription>UPnP-X10 Light and Dimmer
control</modelDescription>
    <modelName>X-10L1</modelName>
    <modelNumber>L1</modelNumber>
    <modelURL> http://iieg.essex.ac.uk/</modelURL>
    <serialNumber>0000001</serialNumber>
    <UDN>valueTObeRetrieved</UDN>
    <UPC>00000-00001</UPC>
    <iconList>
      <icon>
        <mimetype>image/png</mimetype>
        <width>16</width>
        <height>16</height>
        <depth>2</depth>
        <url>icon.png</url>
      </icon>
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:xlighting</serviceType>
        <serviceId> urn:schemas-upnp-
org:serviceid:xlighting:1</serviceId>
        <controlURL></controlURL>
        <eventSubURL></eventSubURL>
        <SCPDURL>testlighting.xml</SCPDURL>
      </service>
    </serviceList>
    <presentationURL>xlighting.html</presentationURL>
  </device>
```

Figure 2. UPnP light device descriptions

4.1.2 A DECONSTRUCTED MODEL – VIRTUAL DEVICE

As devices and their services in ubiquitous environments are discoverable and accessible, a number of possibilities emerge. For instance there is the potential to group communities of services together enabling a community of services to coordinate their actions in a coherent fashion thereby enabling the creation of a "virtual device". This new model of "virtual devices" offers a radical alternative the conventional perception of a "device" (or appliance), as the functional units that make up current devices may be shared. The rationale is that a "virtual device" made up of other devices' functionalities could accomplish tasks, that individual devices were not capable of. For example a media device in the ubiquitous environment could provide 2 services – a file service and a control service. This device would be capable of playing audio independently of other devices. Should a second device, such as a media storage device with audio files be available, a new possibility arises. For instance the media storage device could use the functions provided by the media device for playing its audio files. Thus this new approach changes the traditional model of what an audio player is as, in

[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/

this case, it is a combination of a file server and a media device found in separate appliances. Clearly this is a simple example but the more networked services that are available then the greater the variety and complexity of virtual devices it is possible to create. With this fresh perception a new form of "virtual media device" is born. "Virtual devices" could have an impact on developers on how to develop their products. More importantly, end users could leverage this "device and service rich" ubiquitous environment to create their own "virtual devices" to suit their needs. We refer to such communities of "virtual devices" as **M**eta**App**liance**s (MAPs),** and the representation as the virtual deconstructed device model.

4.1.3 METAAPPLIANCES (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 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 if 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 'Antecedent' and 1-n 'Consequent', and a MAp legally can contain 0-n Rules (as Rules can be added later by the end user). A UML representation of MAp relationships with *Rules* is shown in Figure3.

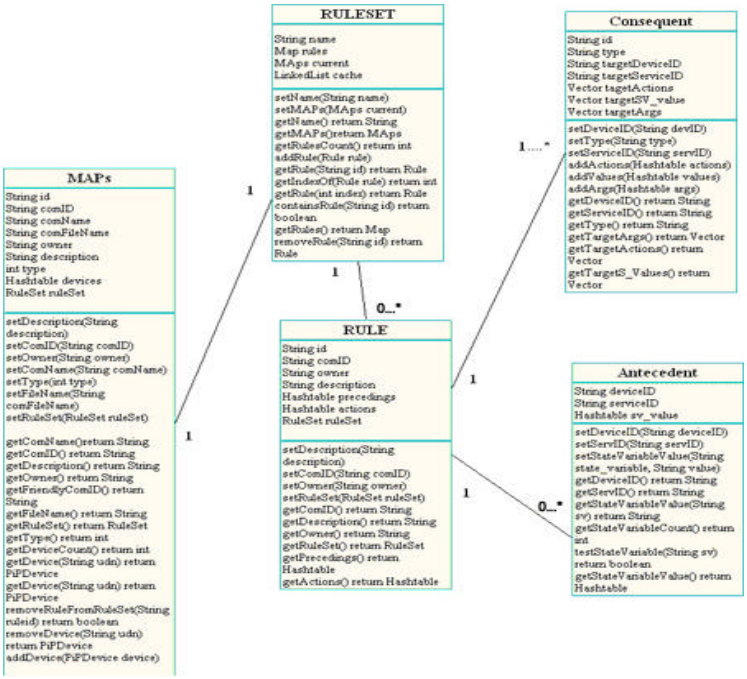

Figure 3 The UML representation of the MAp object structure and its Rules

From the end users' viewpoint a MAp is just a "program" that would create the sort of environmental behaviour they want (eg to accomplish a task or maintain a state for example). As individual end users have their unique preferences and their particular needs, it makes sense to let the users define theirs own MAp (ie their own "program"). Thus *MAps are created under the directions of end users*. As mentioned earlier, MAps can have a graphical representation and thus be visible to the user who created them, either at the time of creation or later when they can be retrieved, edited, shared, executed, or removed on demand.

Lets' begin with a simple scenario to illustrate MAps:

Megan is watching the news on broadcast TV in the lounge when an interesting news item starts. She calls up to John to tune in. John is currently in the office upstairs working on a document and listening to a broadcast radio programme on a hifi system. He starts up the TV application which uses the audio channel of the hifi for output and tunes in to the news item. Once the article has finished John closes the TV application and continues with his work. The hifi switches from the TV audio back to the (paused) music he was listening to.

In the above scenario John's MAp consists of 3 generic "virtual devices", they are: a "TV" device, a "radio" device and a "hifi" device, along with 3 "virtual services": TV-control-service, radio-control-service, and hifi-Control-service. The event: "when John's TV application is on" is the *condition* of a given set of *actions* in MAps. The sequences: "his hifi system halts its current operation and switches its playing channel to TV audio channel and play the audio" are the actions need to be performed if their conditions are met; in this case there is only one condition. A partial definition for John's MAp is showed in Figrue 4.

The differences between a Task and a MAp is that a Task refers to a set of functions (actions) that are required to be performed via a specific command (normally a one-shot sequence) and requires expertise for its definitions whereas a MAp, although it may provide the same functionalities that a Task provides, is an on-going (non-terminating) process and requires no specific expertise for its formation. Also until it is terminated, it will always provide the same functionalities that the user originally created (ie. it is a continually running process). Thus MAps are designed by users to create virtual devices that provide functionalities to customise their personal space in the ubiquitous environment.

```
<com:TransitoryMAp rdf:ID="JohnMAp">
 <com:communityID>Tran-JohnMAp</com:communityID>
 <com:communityName>JohnMAp</com:communityName>
 <com:communityDescription>John testing virtual MAp</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">John</person:firstName>
   <person:nickname rdf:datatype="&xsd;String">Johnny</person:nickname>
   <person:gender rdf:resource="#Male"/>
  </person:Person>
 </com:hasOwner>
<com:hasCommunityDevice>
 <com:CommunityDevice>
<device:deviceUUID>UUID:iPHLDigitalTV17</device:deviceUUID>
 </com:CommunityDevice>
 <com:CommunityDevice>
<device:deviceUUID>UUID:PHLhifiMMS223</device:deviceUUID>
 </com:CommunityDevice>
 <com:CommunityDevice>
<device:deviceUUID>UUID:WonderInternetRadio42</device:deviceUUID>
 </com:CommunityDevice>
</com:hasCommunityDevice>
<rule:hasRuleSet>
<rule:RuleSet>
<ruleSetID rdf:datatype="&xsd;String"> 3e4edfa8-055e-4ef0-8581-70156c156288
</ruleSetID >
<rule:hasRule>
<rule: NonPersistentRule>
<ruleID rdf:datatype="&xsd;String"> ce4edfa8-c55c-4ef9-8581-40156c156258
</ruleID>
</rule: NonPersistentRule>
</rule:hasRule>
</rule:RuleSet>
</rule:hasRuleSet>
</com:TransitoryMAp>
```

Figure 4. A partial definition of John's MAp

4.2 PIP DESIGN RATIONALE

Before we describe the PiP architecture in detail, the following section presents the design rationale underlying PiP.

The specific user population that PiP is designed for can be characterised by the following groups:

- users who wishes to take control of their physical environment
- users who are not reluctant to make programs in their physical environment
- users who wish to use ubiquitous devices to accomplish tasks not available of f-the-shelf"
- users who has little or no programming knowledge
- users who have no expertise in ubiquitous computing.

As PiP targets non technical end-users and leverages end-user programming techniques, its design goals include:

- shielding the user from needing technical expertise (such as ubiquitous networking, information exchange, retrieval, event handling, or general system computation)
- shielding the user from low level programming, algorithms, functions and their abstractions (such as system computation, processes, methods, constants, data structures etc).
- providing a natural platform for the user to create their "programs"
- shielding the user away from the software architecture
- providing the user with as much flexibility as possible while they "compose programs" or MAps
- providing visibility of the user's "programs" or MAps

The system functionality design includes ability to:

- create representations of programs according to the user's example/demonstration
- provide accessible program representations to the user
- present the contents of the program in a meaningful format to the user
- represent the program independently of software architecture
- read the program representation, as well as viewing it during construction
- to save the program or MAps
- to retrieve MAps for later use
- enable MAps to be instantiated (executed) on demand
- enable MAps to be terminated on request
- enable Maps to be amended/edited

4.3 SYSTEM ARCHITETURE

PiP (UK patent No. GB 0523246.7) is deigned to work in real time within a ubiquitous computing environment. 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, where sequences of instructions or actions are significant, PiP assumes the logical sequence of actions is not important. It employs a rule policy to maintain a MAp process in which "a set of conditions is satisfied if the conditions defined within the context of this set are all satisfied". For instance this statement: "if the telephone is ringing and, if the audio is playing, then stop the audio and raise the light level" will have the same logical meaning of the following statements:

- "if the telephone is ringing and if the audio is playing then raise the light level and stop the audio"
- "if the audio is playing and if the telephone is ringing then stop the audio and raise the light level"
- "if the audio is playing and if the telephone is ringing then raise the light level and stop the audio".

PiP leverages UPnP™ technology as its middleware and communication protocol, enabling simple and robust connectivity among devices and PCs. It has modular framework (Figure 5), comprises six core modules as follows:

1. "Interaction Execution Engine" (IEE) – this module has a control point and is responsible for device discovery, service events subscription, and performing network actions requests. It interfaces with low-level network layer via UPnP protocols. The Interaction Execution Engine features a 2-way function that, for in-bound functions, makes calls to the Knowledge Engine Component upon receiving networked events and for the out-bound functions, packages network actions together with internal properties for sending as requests to the network. The Interaction Engine Component also maintains and manages an event-subscription list which it uses to store details of devices whose services are ustilised in MAps. The information required for the event-subscription list is provided by the Event Handler which in turn is driven by requests from the MAp Associator Component trigger by the user's interactions.

Event	Type/Method	Data
DataModel Event	DEVICE_UPDATED	Object
	REMOVE_DEVICE	Object
	STATE_CHANGED	Object
	SERVICE_UPDATED	Object
PiPUIEvent	SET_CURRENT_MAp	Object
	MAp_DELETED	Object
	SET_CURRENT_RULE	Object
	RULE_DELETED	Object
RuleModelEvent	ANTECEDENT_ADDED	Object
	ANTECEDENT _REMOVED	Object
	CONSEQUENT_ADDED	Object
	CONSEQUENT _REMOVED	Object
	ANTECEDENT _STATE_CHANGED	Object
	CONSEQUENT _STATE_CHANGED	Object
	RULE_CHANGED	Object
CCFC Event	DUPLICATES_CONFLIC	Object
	CCFCACTIONS_CONFLICT	Object
	DUPLICATE_ CONSEQUENT	Object
MAp Event	MAp_REGISTERED	Object
	MAp_REMOVED	Object
	REGISTER_DEVICE	Object
	REMOVE_DEVICE	Object
	REMOVE_ALL_DEVICES	Object
	RULE_ADDED	Object
	RULE_REMOVED	Object
PiPUPUPnPService Event	stateChanged(StateVariable variable)	StateVariable

Table 1. PiP Event attributes table

2. "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 function is to communicate events between interested parties. Interest parties would need to register with the Event Handler in order to receive appropriate events and their data. The table shows in Table 1 illustrates the types of events and the data handle by the EH.

3. "Knowledge Engine" (KE) – this module is responsible for assembling and instantiating a "virtual device" before storing them on to the Knowledge Bank. It is also responsible for updating the device' current status as well as maintaining an up-to-date content of the knowledge bank. The KE Component notifies EH when to send a "DataModel Event" (seeTable 1) to interested parties as a result of any change to the Knowledge Bank.
4. "Real-time MAp Maintenance Engine" (RTMM) — is a process that maintains the records of current and previously created MAps. The MAp Associator registers (with the EH) for GUI based user interaction action events. For example when the user "drags & drops" devices into a "programming formation palette", the GUI component notifies the EH which generates a related event, which, in turn, is passed to the MAp component. Likewise, the MAp Associator Component notifies the EH of any changes in the user's MAp which, in turn, generates a MAp Event which is sent to interested parties.
5. "Real-Time Rule Formation Engine" (RTRF) – this module is responsible for assembling rules based on user interactions within the "demonstration" mode (starts when user clicks "ShowMe" button; ends when user clicks "Done!" button). This component registers with the EH for KE, MAp events and GUI events. The RTRF Engine "listens" for user's activities which are communicated to it via the events sent by the EH. Having realised the user's direction via GUI events, it "understands" the user's activities and generates appropriate required fragments according to the user's directions with the events received from the EH before assembling them to the rule and pass it on to RTMM engine.
6. "GUI" – A graphical interface called "PiPView" (Figure 6 and 7) that the user can use to make inspections of the environment, compose/delete Maps/Rules etc, interact with the system and control physical environment. The GUI utilises "drag and drop" or "clicking" as the main modes of interaction and is built around a multi-threaded approach. The main class for the GUI component is the editor class, which represents the user's entry point to the system. In addition there are the eight pop up dialogs and a device control panel which make up the remaining core classes. Helper classes generally provide support functions such as "drag & drop" and screen rendering. To enhance the intuitive nature of PiP, visually, the editor has been designed to convey the familiar "look and feel" of current PC applications. Thus it uses conventional menus, menu-items and tool bar icons to represent the internal functionalities. A "prompt for user's attention" policy is used for global error handling.

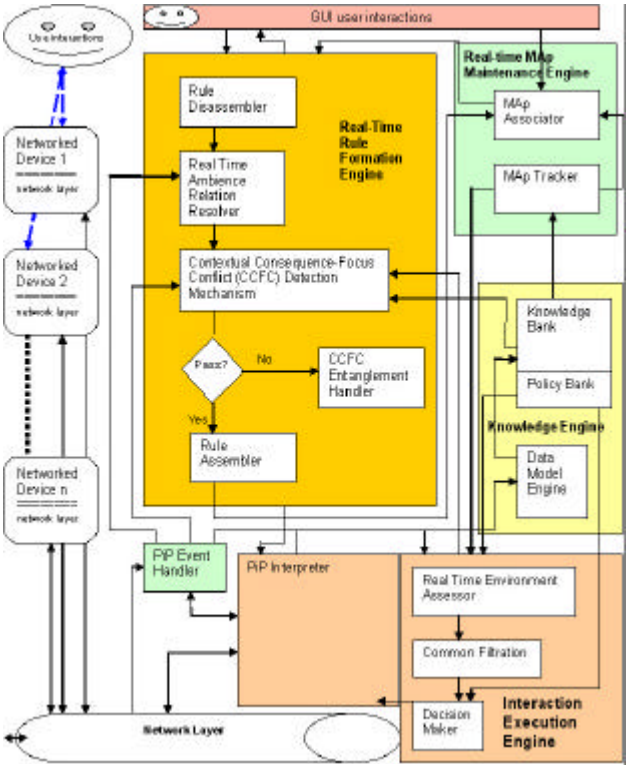

Figure 5. PiP modular framework

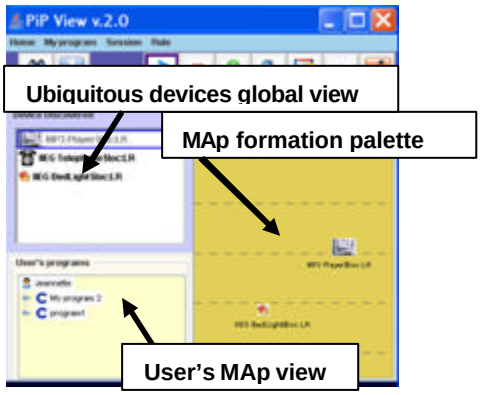

Figure 6. PiPView screen shot1 – user's MAp view

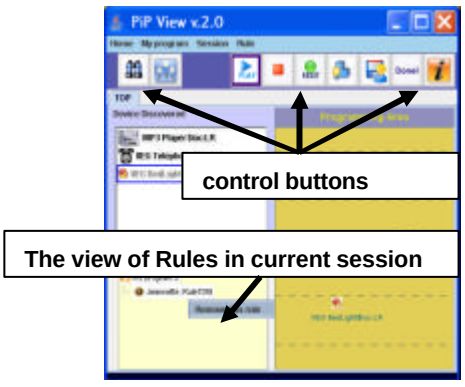

Figure 7. PiPView screen shot2 – Rules view

4.4 HOW THE SYSTEM WORKS?

This section illustrates how the system actually works when put together. In PiP, apart from the GUI (PiPView) mentioned above, the networked items that exist in the ubiquitous environment can be regarded as user interfaces too, as users interacts with them during the demonstration process. Sample interfaces are: networked dimmer lights, networked telephone, network entertaining systems, network fridge etc (Figure 8). Through these user interfaces, users can interact intuitively and naturally with the environment and the metaphor for configuring their personal space is very simple: the user just needs to create a MAp and show the system the functionalities that the MAp should have by interacting with the environment and PiP will do the rest for the user.

Figure 8. PiP meets ubiquitous environment

To create a MAp the user needs to log-in to the system through PiPView. An instance of the Interaction Execution Engine (IEE) module performs network discovery cycle and reports available devices to the Knowledge Engine (KE), which transforms the data retrieved from the discovery process into appropriate structure and informs the middleman – EventHandler (EH) by passing appropriate parameters ready for retrieval. The EH receives the notification and passes it on to the interest parties, which have been registered at system startup time and are stored in a local list. As one of the interested parties, PiPView receives the notification, an instance of the interpreter helps to transform and render the data into higher level descriptions that are presented to the user who then decides on which devices to use for creating a new MAp.

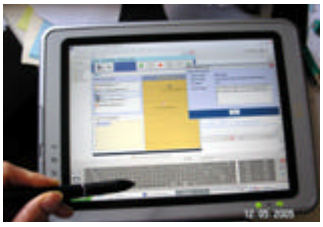

Figure 9. PiP on tablet view

MAp are created by the user "dragging & dropping" device representations into a "formation palette" which defines the set of devices that, via the EH and IEE, may share events. When a set of MAp devices are selected (the user is free to save the MAp at any time during her demonstration with a meaningful name. The user can add or remove selections at any time during this period) the MAp is given behaviour by user engaging physical activities on the real system, via PiPview, when to the process of showing the system behaviour will begin which, in turn activates the Real Time Rule Formation (RTRF) Engine. In the mean time, the user demonstrates actions on how the meta-appliance should behave by providing examples using any of the three methods: (1) physically interacting with the devices themselves, (2) using a UI control panels (3) a combination of both, where the choice is left to the user. Based on the actions the user demonstrates, the devices generate appropriate events for the network. The RTRF Engine "listens" for user's activities which are communicated to it via the EH (which, in turn, is informed by the KE when it receives the status of the remote devices have changed). This behaviour is encoded as a set of rules composed of two parts; an antecedent (conditions) and consequent (resulting action) as the RTRF Engine encodes " the user's demonstrated actions.

Figure 10. End-User programming via physical interaction

In PiP the user is given as much freedom as possible, allowing antecedent 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 facilitate this, a Rule Dissembler is employed to separate antecedents and consequents. The rules are shaped by the Real-Time Ambience Relation Revolver which is responsible for resolving the relationships of the information received from the network's internal system and the user. As part of this a Contextual Consequence-Focus Conflict (CCFC) Detection Mechanism tests the user's current action to see if there is a conflict that may result in unwanted system behaviours based on the existing MAp rules. Finally, if the CCFC detects no conflicts then the Rule Assembler constructs the completed rule. Otherwise, the CCFC Entanglement Handler will deal with this situation by performing the following operations (1) gathering the conflict information (2) presenting conflicts to the user while (3) isolating the conflict actions from those in the current

MAp. To execute a MAp, the user simply needs to drag the MAp graphical representation from the user's home program area (represented in a sub-view) dropping it on the "play" button located at the top of the PiPView. To terminate a MAp, the user simply clicks the "stop" button.

5.0 RELATED WORK

This research area is receiving g a great deal of attention although most of the research and tools are aimed at developers rather than end users. With respect to digital homes, which our work is directed at, the main trends there are three main categories of related work:

5.1 END USER COMPUTING

Visualisation techniques are often employed for making computer systems more comprehensible and easy to use. For example some X10 clients provide graphical interfaces that allow users to specify the behaviour of various devices or objects in their homes based on events or conditions [X10 Client]. SiteView [Beckmann03], has incorporated tangible techniques for programming active environments with predictive visualizations. The Speakeasy system [Edwards02] supports the *ad hoc*, end-user configuration of devices and applications through a set of patterns defined in mobile code via web interfaces. Research has extended information interfaces to physical mechanisms such as Ubi-Finger, a gesture-input device developed by [Tsukada02], which uses sensor techniques together with an infrared transmitter allowing the user to control an ubi-device by first pointing at it and then using a finger gesture. In addition, the iCAP system [Sohn03] allows users to prototype context-aware applications rapidly using a pen-based interface for specifying input and output devices, as well as behavioural rules through 'drag and drop' interactions and pie menus. In general, end-user computing developments offer users alternatives on how to use and customize ubiquitous applications but they have not developed enough to offer non-technical users the capability/ flexibility to design and make "programs" in their own user defined environments.

5.2 TASK COMPUTING

Task computing is a variant of end-user computing. Its main goal is to abstract application functionalities or services, along with associated attributes, into "Tasks" ie. high-level goals people want to achieve, by adding an additional layer of abstraction over the underlying service or device functionality layer. The task abstraction seeks to represent a user's computing intention and initial developments in this area were aimed at dynamic configuration of composite services eg a "lecture presentation" task might be the coordination of services that switched on the projector, lowered a screen and dimmed the light. In this example tasks can be seen to have a macro like nature. Task driven computing, was proposed by Wang and Garlan [Wang], aimed at helping travellers automate their service configurations in different environments. The AURA [Garlan et al] project aimed at constructing a global environment where task-based configurations were mobilised and automatically restored within all environments visited by a user. Semantic "Task Computing", developed by Fujitsu Laboratories [Masuoka03, Masuoka04] took this concept forward significantly by enabling end-users to perform more complex tasks in ubiquitous environments. A variant of this concept called the Personal Operating Space (POS) was explored by Shahi [Shahi05]. In this approach the user was imagined to have a generic set of preferences that described the users favoured configuration for every environment visited including both information (eg PC work sessions and tools), and physical spaces (eg building services). The system was implemented using a mobile phone as the container of personal preferences and was characterised by the slogan "the phone is me". From a computing perspective, Task Computing has the advantage of eliminating the need for the user to know how to achieve the end results, thereby allowing them to focus on the results which in turn has led to claims of significant productivity gains compared to conventional approaches [Wang]. However, there remain a few issues to be addressed for this approach to be more widely adopted. For instance the tasks themselves need to be created in advance and, given the ambiguous nature of the abstractions which form the basis of the Tasks, a degree of guesswork on what users might need and technical expertise is required for their definition. To date end users, have not been able to design and customise their own tasks, instead users use the pre-defined tasks built by experts.

5.3 END USER PROGRAMMING

End-User programming is characterised by the use of mechanisms to allow non-technical end-users to create "programs". Earlier work in this area was targeted primarily at desktop computing. Recently this vision has found its way into technology-rich ubiquitous environment where a number of different approaches are being explored. For instance, Humble [Humble et al] use a jigsaw, metaphor, enabling users to "snap" together puzzle-like graphical representations as a way of building applications. The HYP system [Barkhuus03] enables users to create applications for context-aware homes using a mobile phone based graphical interface. Media Cubes [Hague03] offers a tangible interface for programming an environment in which each face of a cube is represented by a set of program structures. In this approach

"Programming operations" are achieved by turning the appropriate face of the cube towards the target device. Truong's CAMP project [Truong et al] 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 end-users to realize context-aware ubiquitous applications in their homes. The Alfred project proposed the concept of "goals", and "plans" to allow users to compose a program via "teaching-by-example". The system utilised a macro programming approach which could be created by the user via verbal or physical interactions. This differs significantly to PiP which spawns non-sequence dependent processes based on a virtual device metaphor rather than macros and a procedural programming metaphor. According to 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. [Gajos02] In summary, end-user programming is a vibrant area of research with a number of excellent and contrasting approaches being researched

5.4 SUMMARY

Most research to-date has aimed at simplifying the process for creating or customizing ubiquitous applications. PiP is positioned firmly in the area of end-user programming in which most approaches currently utilises graphical interfaces and macros. Whilst macros are simple and intuitive, their dependence on strict order makes them fragile and susceptible to failure in situations where order is not fixed; hence PiP does not use sequence dependent processes. Furthermore, the avoidance of macros also provides greater flexibility, freeing end-users from the need to follow strict instructions. Likewise, whilst procedural programming constructs are intuitive to computer scientist this is not necessarily the case for non-technologist. Thus in PiP we have opted for a metaphor commonly used in the home, the appliance, enabling the end-user to create her own 'virtual devices' via simple interaction with the environment. As explained earlier in this paper in PiP we decided not to use macros so we would avoid the problem of generating fragile solutions that were dependent upon a logical sequence of behaviours (eg. the program will fail if the sequence it depends on break).

6. EVALUATION

We have developed PiP to the point where an evaluation equivalent to a proof of concept could be carried out. This section presents the evaluation and the results gathered. The methods used in the evaluation were qualitative and included interviews, observation and video recordings.

6.1 TEST BED – THE iDORM2

The evaluation was carried out in iDorm2[2], a newly built two-bedroom flat at the University of Essex (Figure 11) to be an experimental ubiquitous computing environment. A common interface to the iDorm2 and its devices was implemented through Universal Plug and Play (UPnP) which is an event-based communication middleware that give devices the ability to automatically discover each other and communicate.

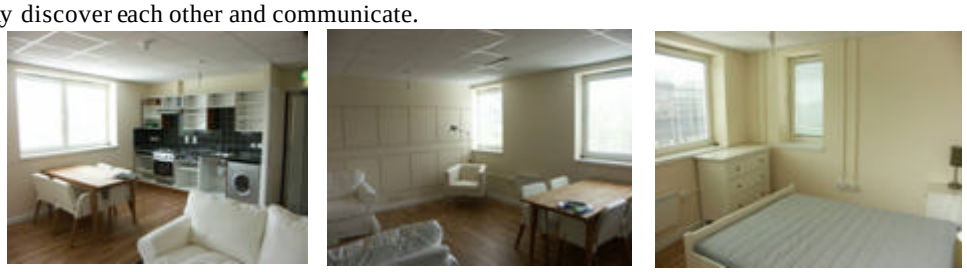

Figure 11. The iDorm2 test bed

Five networked devices were created for the user evaluation; these were: (1) a bed light , (2) a desk-light, (3) telephone, (4) a sofa and (5) an MP3 player. The telephone and the sofa contained embedded sensors that reported on their status. The two lights had sensors and actuators to switch them on/off and report their status. These four devices were connected to a snap [Snap] board which provided a UPnP based network interface. The fifth device, an MP3 player, was implemented as a software emulation running on a laptop with a mouse interface. A gateway

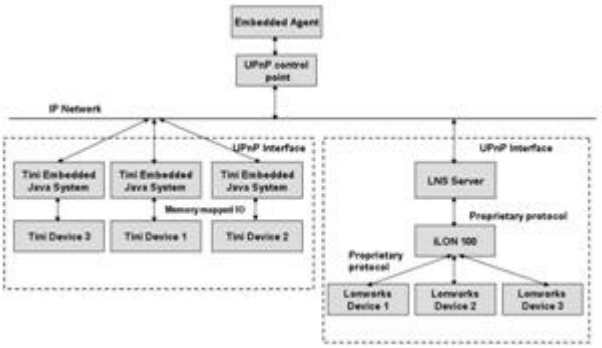

Figure 12. The iDorm2 logical network infrastructure

[2] iDorm2 at http://iieg.essex.ac.uk/idorm2/index.htm

server was used to run proxy UPnP software devices. Figure 12 shows the logical network infrastructure of iDorm2.

6.2 EVALUATION DESIGN AND PROCEDURE

The end-user evaluation was designed to appraise PiP. Our objectives for the evaluation were to (1) assess whether PiP met its design objectives (2) assess how easy or not the end users found PiP to be in customising their personal space and (3) gather a view from a broader background of PiP users. For this latter test we solicited assistance from social scientists and devised a six dimension assessment comprising: the overall concept, user controls, cognitive load, information retrieval/visualisation, affective experience and the future. Our strategy was to set-up open ended trials giving the end-users as much freedom as possible so that we could get a better idea of how participants would like to use the system, and how the system coped with different users. User freedom included time, methods, and tasks.

6.2.1 PARTICIPANTS

We invited eighteen participants drawn from a diverse set of backgrounds (eg housewives, students, secretaries, teachers etc). There were 10 females and 8 males and their ages ranged from 22 to 65. The participants formed a multicultural group including Asians, Europeans, Hispanic and Australians. All participants had some minimal computing experience ie. they knew how to use a mouse; see Figure 13. Whilst 21.3% of the participants had a very good knowledge of programming, 57.4% of them had none at all; see Figure14.

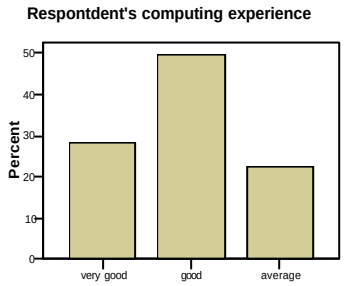

Figure 13 . Participant's Computing Experience

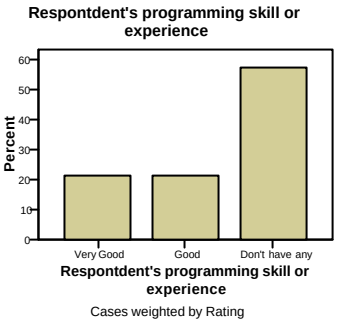

Figure 14. Participant's Programming Experience

6.2.2 DESIGN OF QUESTIONNAIRE

The questionnaire was developed to assess the participants' subject judgements about the usability of PiP. It consisted of a set of seventeen statements, measuring attitudes over six usability dimensions: "Conceptual", "User Control", "Cognitive Load", "Information Presentation", "Affective Experience" and "Future Thoughts". The questionnaire used a five-point Likert scale with responses from "Strong Agree" through to "Strongly Disagree". Each of these dimensions consisted of a series of statements (from 2 to 4) and each statement offering a range of ratings (from 1 to 5). A higher rating score on the dimensions contributes towards the greater usability of PiP. The questionnaire was then iterated for checks and revised any ambiguous statements which may not be clearly understood by the users before it was piloted on 3 users. In papers there is some discussion as to how to best construct this type of test with, for example,, some researchers worrying that there is no metric, interval measure or that the data would be best treated as ordinal [Coolican 94]. However, there is a widely accepted consensus that the Likert scale can use with interval procedures, provided the scale item has at least 5 and preferable 7 categories [Oppenheim92]. Thus, as we are using 6 categories, the questionnaire rating data was treated as interval data in this study.

6.2.3 PROCEDURES

Prior to the evaluation, a consent form was prepared and completed by all participants before commencing their sessions. This measure was undertaken to ensure that the participants were fully aware of the type of data that would be collected during the session and its use after the session was completed.

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. To support the evaluation a "user-action" module was created and installed in PiP to collect system data. A digital video recorder was used to record participants' interactions and verbal comments.

Figure 15. Screenshots of participants in trials

Each trial was preceded by a 20-minutes training session to allow participant to familiarise themselves with the system, the nature of the task and the environment. This 20-minutes training session included: a briefing on the PiP concept, a walk through using the PiP UI, a quick demo by-example on how to compose a MAp followed by an introduction to the trial environment. The task for the trials was that the participant should use PiP to customise their personal space in the way they wanted; no specific type of behaviour for the environment was set for trials, rather, the participants were free to create one or more MAps of their own. After creating MAps participants were encouraged to switch between the MAps they had created.

No time limit was set for the participants to customise the space. Assistance was provided where needed. Following completion of the evaluations, a questionnaire with 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 dimensions mentioned above. Data was analysed using SPSS.

6.3 RESULTS

6.3.1 PERFORMANCE

After the brief 20-minutes training session, we found that 83% of participants were able to use PiP to customise their personal space with little or no assistance. The time taken to accomplish these tasks varied from participant to participants but our evaluation objectives didn't include measuring the time taken. Of the two methods available for demonstrating examples 11% of the participants chose to customise their personal space wholly via GUI controls but 72% of them conducted by physical interactions with the environment while the rest used a mixture of both. There was no sign of distress shown during or after the evaluations. Although PiP is not exacting on logical sequence when composing MAp, 33% of the participants expressed the view that they found it mentally easier using a logical sequence and decided to conduct their trials that way. The remander of the participants (77%) focused on the task (ie creating the behaviour of the environment rather than logical sequence). The study also revealed that none of the participants found it difficult to understand the basic principles of the system. During the evaluation no network problem was encountered and the system behaved and acted correctly in all cases.

6.3.2 QUESTIONNAIRE RATING

Various tests were carried out to analyse the questionnaire ratings using SPSS software package. Table 2 shows an overall rating scale for six dimensions evaluated. From the results we observed that "Affective Experience" dimension received the highest rating. 148 out of the total number of 240 cases received a top rating, which is 61.7%. The "Information Retrieval" dimension - information presentation - had the lowest recorded rating (2) whereas in all other dimensions 3 was the lowest recorded. Tests also revealed that the overall difference between the lowest (4.1) and highest (4.6) mean ratings was not great (see Table 2). The highest mean rating was scored by "Affective Experience" dimension suggesting participants were enjoying the experience of programming using PiP while the lowest was found directing at the "Future Thoughts" dimensions. The cognitive load dimension had an overall average score of 4.3 indicating participants found the process relatively simple. From individuals' tests we observed that an overall 83.4% of all participants found PiP intuitive to use and 94.4% of all participants stated they felt the experience rewarding.

	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 2. One-Way ANOVA test on dimension vs qRating

We also completed a cross analysis based on the participants computing and programming experience ranging from average, good to very good each experience respectively (Figure 16). Due to the length limit of the paper, only the results of the group with average computing and programming experience are reported here.

For this group of participants, 4 out 199 cases evaluated had negative responses (2%). However, the overall results showed that they rated highly for all six dimensions (Figure 16). Remarks recorded from this group included: "I just feel like right now I want to sit down for a lot longer and try out all sorts of environment that I could possibly create!" and another one : "I can really get quite keen on it" were just the few examples of the positive subjective judgement.

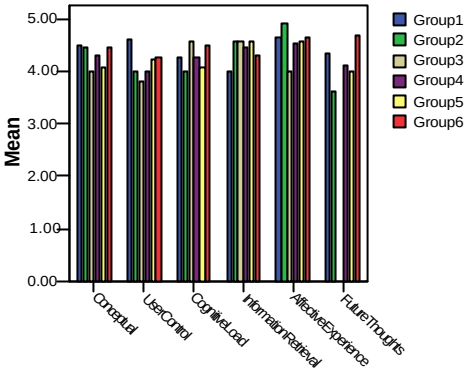

Figure 16. Mean ratings bar chats on six dimensions for each individual group of participants.

7. CONCLUSION AND FUTURE WORK

This paper has described our research into an end-user tool for customising personal space in ubiquitous environments. We have successfully implemented a "proof of concept" system called PiP, using an event-based modular architecture design which enables non-technical end-users to create MAps (ie. a "program" based upon end-user preferences), for customising their personal space in ubiquitous environments. The end-users were not required to follow a rigid logical sequence when customising their personal spaces, although we observed a minority of the participants, notably those with a computing background, preferred using a logical sequence.

We also reported on our end-user evaluation results based on a group of 18 participants selected from different backgrounds to evaluate their subjective views on the value of the research. Whilst we acknowledge that the participants are only a minute sample from the population, the initial results are encouraging as they show that PiP served different users well, allowing them to customise their personal space. We found that the usability dimension measures returned a mean of at least 4 suggesting the participants found PiP to be both useful and enjoyable. Based on the results gathered, we concluded that PiP was able to support users composing MAp for customising their personal space in ubiquitous computing environments. At the outset of our work, one of our contentions for PiP was that participants would find the manual experience of customising their personal space relatively easy. This assertion was supported by the results as, the cognitive load dimension had an overall average of 4.3, indicating in the views of the participants this process was relatively simple. Thus, we contend that whilst the user evaluation was a relatively small proportion of the population, it has provided evidence that this approach is usable by non-technical end-users to create their own functionalities in ubiquitous computing environments, such as digital homes. In addition, since the system operations are determined by MAps which are created under the direction of end users, we argue that PiP provides more operational transparency and control, thereby addressing the ethical and human value concerns of those who worry about "who, what and when" access to systems in there environments.

For our future work we hope to conduct further work on knowledge representation at the MAp level. 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. For this we propose to look at ontology's such as OWL-s and dComp[3] [Chin05a, Chin05b] (the latter being an ontology we have developed and published on elsewhere). In this paper we have identified other interesting approaches to supporting end-users in pervasive environments and another goal we have is to examine how our work might support some of these paradigms. In particular, the work on Task Computing, which currently lacks an end-user mechanism for enabling end-user to form task might usefully be able to use PiP. Finally, we would like to conduct a larger scale evaluation involving more users and more complex tasks. This will require us to build more networked devices and find more users. In the fullness of time we look forward to reporting on all of these developments. Finally, we hope that PiP can move "Do it Yourself" from the realm of "wallpaper and paint" into "electronics and computing" for the non-technical ubiquitous home based user.

[3] dComp at : http://iieg.essex.ac.uk/dcomp/ont/dev/2004/09/

Presented at the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06), Wuhan and Three Gorges, China, September 3-6, 2006

ACKNOWLEDGEMENTS:
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 there support in both scientific and personal terms.

REFERENCES:

[Andersen 05] Anderson,P.; Bardram, J.E.; Christensen, H.; Corry,A.V.; Greenwood, D.;Hansen, KM.; Schmid,R. *"An Open Architecture for Palpable Computing*", Proceedings of the Object Technology for Ambient Intelligence Workshop (OT4AmI).

[Ballagas 03] Ballagas, R., Ringel, M., Stone, M., and Borchers, J.: "*iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments"*, Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2003). ACM Press, New York (2003) 537-544

[Barkhuus03] Barkhuus, L., Vallgårda, A: "*Smart Home in Your Pocket*", Adjunct Proceedings of UbiComp 2003 (2003) 165-166

[Becker et al] 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.

[Beckmann 03] Beckmann, C., Dey, A."*SiteView: Tangibly Programming Active Environments with Predictive Visualization*", adjunct Proceedings of UbiComp 2003 (2003) 167-168

[Callaghan 05] Callaghan V,; Colley M,; Hagras H,; Chin J,; Doctor F,; Clarke G,"*Programming iSpaces: A Tale of Two Paradigms, in book Intelligent Spaces"*, The Application of Pervasive ICT part of the series Computer Communications and Networks, Steventon, A; Wright, S (Eds.) approx. 455 p. 162 illus. ISBN:1-84628-002-8, Dec 2005.

[Chin 04] Chin JSY.; Callaghan V.; Clarke G.; Colley M,; Hagras H. "*Pervasive Computing and Urban Development: Issues for the Individual and Society*", UN Second World Urban International Conference on The Role of Cities in an Information Age, Barcelona, Spain, 13-17 September, 2004.

[Chin 05a] J, Chin , V. Callaghan, M. Colley, H. Hagras, G. Clarke, "*Virtual Appliances for Pervasive Computing: A Deconstructionist, Ontology based, programming-By-Example Approach*", The IEE IE05, Colchester, UK, 28-29 June 2005.

[Chin 05b] Chin J, Callaghan V, "*End-User Programming in Pervasive Computing Environments"*, The International PSC-05, Monte Carlo Resort, Las Vegas, USA, June 27-30, 2005

[Cook 04] *D. J. Cook, S. Das, "MavHome: Work in Progress"* IEEE Pervasive Computing, *2004.*

[Coolican 94] Coolican H. "*Research Methods and Statistics in Psychology*" (2nd Edition) Hodder and Stoughton, 1994.

[Cypher 93] 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.

[Drossos 05] Drossos N., Goumopoulos C., Kameas A., "*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

[Edwards 02] Edwards, W.K., Newman, M.W., Sedivy, J., Smith, T., Izadi, S.: "*Challenge: Recombinant Computing and the Speakeasy Approach.*" In: Proceedings of the Eighth Annual International Conference on Mobile Computing and networking (MobiCom 2002). ACM Press, New York (2002) 279-286

[Gajos 02] Gajos K., Fox H., Shrobe H., "*End User Empowerment in Human Centred Pervasive Computing*", Pervasive 2002, Zurich, Switzerland, 2002.

[Garlan 02] Garlan, D.; Siewiorek, DP; Smailagic, A; Steenkistie, P; "*Project Aura: Toward Distraction-Free Pervasive Computing*", IEEE Pervasive Computing Magazine, April-June 2002.

[Gellersen 02] Gellersen, H.W., Schmidt, A., Beigl, M."*Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts"*, Mobile Networks and Applications (MONET). 7(5)(2002) 341-351

[Greenberg 01] Greenberg, S. and Fitchett, C.: Phidgets: Easy Development of Physical Interfaces through Physical Widgets. In: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST 2001). ACM Press, New York (2001) 209-218

[Guibert 03] Guibert N. Girard P.,"*Teaching and Learning Programming with a Programming by Example System*", Int'l Symp on End User Development, Sankt Augustin (Bonn), Germany, 2003

[Grimm 01] Grimm, R.; Davis, J.; Lemar,E.;Macbeth,A; Swanson, S.; Anderson, T.; Bershad, B.; Borriello, G.; Gribble, S.; Wetherall, D, " Programming for Pervasive Computing Environment", Proceedings of 18th ACM, Symposium on Operating System Principles, Canada, Oct., 2001

[Hague 03] Hague, R., Robinson, P., Blackwell, A.: Towards Ubiquitous End-user Programming. In: Adjunct Proceedings of UbiComp 2003 (2003) 169-170

[Humble 03] 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

[Holmquist 01] Holmquist, L.E. et al " Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artifacts", Proceedings of Ubicomp 2001, Atlanda, GA, USA, September 2001.

[IIEG] The intelligent inhabited environment group http://iieg.essex.ac.uk

[**Ishii 04**] Ishii, H., Ratti, C., Piper, B., Wang, Y., Biderman, A., Ben-Joseph, E. Bringing clay and sand into digital design - continuous tangible user interfaces, BT Technology Journal, Vol. 22, No. 4, October 2004, pp. 287-299

[Kameas 03] 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.

[Klemmer 04] Klemmer, S.R., Li, J., Lin, J., Landay, J.A.: Papier-Mâché: Toolkit Support for Tangible Input. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2004). ACM Press, New York (2004) 399-406

[Limb 05] P R Limb, S. Armitage, JSY Chin, R Kalawsky, V Callaghan, P M Bull, H Hagras, M Colley, "*User interaction in a shared information space – a pervasive environment for the home"*, Perspectives in Pervasive Computing, 25th October 2005 – IEE, Savoy Place, London

[Masuoka 04] Masuoka, R., Labrou, Y., and Song, Z.: Semantic Web and Ubiquitous Computing - Task Computing as an Example - AIS SIGSEMIS Bulletin 1(3) October 2004.

[Masuoka 03] Masuoka R, Parsia B, Labrou Y, *"Task Computing - the Semantic Web meets Pervasive Computing*," 2nd Int'l Semantic Web Conf (ISWC2003), 20-23 Oct 2003, Florida, USA

[Mozer 98] 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

[Myers 90] Myers B.A., *"Creating user interfaces using programming by example, visual programming, & constraints"*, ACM Trans Programming Languages & Systems (TOPLAS), Vol 12, Issue 2 (April 1990), pp 143 – 177

[Nagel 01] Nagel, K., Kidd, C.D., O'Connell, T., Dey, A.K., Abowd, G.D.: The Family Intercom: Developing a Context-Aware Audio Communication System. In: Proceedings of Ubicomp 2001: Ubiquitous Computing. Springer-Verlag, Berlin Heidelberg New York (2001) 176-183

[Nichols 01] Nichols J, Myers B, Miller R, Personal Interfaces in Ubiquitous Environments, The CHI 2001 Conference on Human Factors in Computing Systems, Seattle, Washington, 31 March-5 April 2001

[Oppenheim 92] Oppenheim, A N. "Questionnaire Design, Interviewing and Attitude Measurement" Pinter Publishers Ltd, 1992

[Shahi 05] Shahi, A., Callaghan, V., Gardner, M.: Introducing Personal Operating Spaces for Ubiquitous Computing Environments. Pervasive Mobile Interaction Devices 2005 (PERMID 2005), hosted by 3d International Conference on Pervasive Computing, Munich 8-13, May, 2005.

[Smarthome] Smarthome X10 Kit. http://www.smarthome.com

[Smith 77] Smith, D. C., "*Pygmalion: A Computer Program to Model and Stimulate Creative Thought"*, Basel, Stuttgart, Birkhauser Verlag. 1977.

[Snap] Snap http://snap.imsys.se

[Sohn 03] Sohn, T., Dey, A. K.: iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. In: Extended Abstracts of ACM Conference on Human Factors in Computing Systems (CHI 2003). ACM Press, New York (2003) 974-975

[Stevenson 03] Stevenson G, Nixon P, Ferguson RI, A General Purpose Programming Framework for Ubiquitous Computing Environments, In System Support for Ubiquitous Computing Workshop at UbiComp, 2003

[Tandler 01] 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

[Truong 04] Truong, KN.; Huang, EM; Abowd, GD; " CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home", Proceedings of Ubicomp 2004, pp 143-160.

[Tsukada 02] Tsukada, K. and Yasumura, M.: Ubi-Finger: Gesture Input Device for Mobile Use, Proceedings of APCHI 2002, Vol. 1, pp.388-400

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

[Want 95] R.Want, B.Schilit, N. Adams, R.Gold, K.Petersen, J.Ellis, D. Goldberg, M. Weiser, "The parctab ubiquitous computing experiment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, 1995.

[Weiser] Weiser, M. "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, 36(7), pp75-84.

[X10 Client] X10 Client. http://x10controller.sourceforge.net/X10Controller/X10Client/

