In the handbook of Ambient Intelligence and Smart Environments, Editors: H. Nakashima, J. Augusto, H. Aghajan, Springer, 2010, Spring, pp 371-407, ISBN 978-0-387-93807-3

End-user Customisation of Intelligent Environments

Jeannette Chin, Victor Callaghan & Graham Clarke
University of Essex
[jschin] [vic] [graham] @essex.ac.uk

1. Introduction

One of the striking aspects of world-wide-web is how it has empowered ordinary non-technical people to participate in a digital revolution by transforming the way services such as shopping, education and entertainment are offered and consumed. The proliferation of networked appliances, sensors and actuators, such as those found in digital homes heralds a similar 'sea change' in the capabilities of ordinary people to customise and utilise the electronic spaces they inhabit. By coordinating the actions of networked devices or services, it is possible for the environment to behave in a holistic and reactive manner to satisfy the occupants needs; creating an intelligent environment. Further, by deconstructing traditional home appliances into sets of more elemental network accessible services, it is possible to reconstruct either the original appliance or to create new user defined appliances by combining basic network services in novel ways; creating a so called virtual appliance. This principle can be extended to decompose and re-compose software applications allowing users to create their own bespoke applications. Collectively, such user created entities are referred to as Meta –appliances or –applications, more generally abbreviated to MAps.

Deconstruction and user customized MAps raise exciting possibilities for occupants of future intelligent environments, and sets significant research challenges [Chin 09]. For example, how can MAps be constructed and managed by ordinary non-expert home occupants? At one extreme it is possible to use artificial intelligence (AI) techniques and equipment, such as autonomous intelligent agents. These monitor an occupants habitual behaviour, modelling their behaviours, and creating rule-based profiles (self-programming) so they can pre-emptively set the environment to what they anticipate the user would like [Augusto 06]. However, some people have privacy concerns about what is being recorded, when it is being recorded and to whom (or what) any information is communicated. These concerns are particularly acute with autonomous agents, in which people have little direct control. Such matters are especially sensitive when the technology is used in the private space of someone's home. Frequently, end-users are given very little choice in setting-up digital home technology and are obliged to accept whatever is offered [Callaghan et-al 08]. Apart from the issues of privacy and trust, we argue that creativity is an essential and distinctive human quality, and that many people would enjoy the process of creating their own novel networked appliances and personalising their 'electronic spaces', providing they can be shielded from unnecessary technical complexity. This has parallels to the common practice of people decorating their own homes with paintings, walls hangings, pictures, colour schemes and furniture. This rationale has led many researchers to investigate what is termed 'end-user programming', a methodology aimed at allowing non-technical people to personalise their own digital spaces with network enabled embedded-computer based devices. Historically, programming has only been accessible to well-qualified professionals, such as computer scientists, or the outcome of self-programming (learning) using autonomous intelligent agents. The challenge for achieving an end-user programming vision is to devise programming methodologies that are usable by non-technical people.

In this chapter we begin by reviewing current research into end-user programming systems, especially those for the home. We describe approaches that range from transposing conventional programming constructs into graphical or physical iconic objects, to those that adopt radically new programming metaphors. By way of an example of these new approaches, we describe a novel end-user programming approach developed at the University of Essex called Pervasive-interactive-Programming (PiP) (UK patent No. GB0523246.7) and a service coordination model known as Meta-Appliances/Applications (MAps). We report on an evaluation of user experiences using PiP in a digital home, the University of Essex iSpace. We conclude this chapter by reflecting on the main findings of our work.

2.0 The Home of the Future – A Vision

One vision for the home of the future is that it will be will be a technology-rich environment containing tens or even hundreds of pervasive network based services, some provided by physical appliances within the home, others by external service providers (see Figure 1). It will be centred on the concept of aggregating network services to satisfy particular needs. A supporting middleware will make these services discoverable, and accessible to the user in the environment in which they reside. Services might range from simple video entertainment streams through to complex home energy management packages.

Figure 1 – A Pool of Service Providing Appliances

In the current market appliances are designed, packaged and marketed by commercial companies who bundle together pre-formed packages of functions that they anticipate customers will want (e.g. a TV). However, networked technology enables the development of alternative approaches. For example, from a customer's perspective, it might be possible to create network based appliance or environment behaviour by aggregating coordinating sets of networked services. Formal descriptions of these composite services, and their behaviours would form 'soft-objects' which could migrate with people as they move across differing environments (e.g. via the network, or contained in mobile phones), instantiating these functions from local resources wherever possible. The current appliance market suggests that there is a basic set of common needs for devices that people want (eg telephones, TVs,

heating etc) and these could form default MAps in all homes. However other, more novel, MAps - composite service descriptions - created by lay-people could be developed for personal use or even traded. Clearly, the deconstructed model, represents a radically different way of providing appliances and it is not yet clear, that the market would accept such a proposition. However, even if this vision for 'virtual appliances' does not find favour in the market, it remains an alternative way for end-users to personalise the functionality of their digital homes.

3.0 Contemporary Work

Various approaches to customising digital homes have been reported by researchers, ranging from intelligent agents through to user driven methods. A common approach to endowing network appliances with coordinated behaviour is via rules [Callaghan et-al 04]. Rules are fundamental to determining how a given appliance or service interacts with other appliances or services [Zamudio et-al 08]. Chin [Chin et-al 08] has proposed a taxonomy based on '*rule formation'* as a way of describing the different rule-based approaches to customizing digital homes. Chin's taxonomy is based on the following categories:

1. *Pre-programmed rules* - usually created by the developers or manufacturer,
2. *Agent-programmed rules* - created from intelligent agents, artificial intelligence or machine learning mechanisms, and
3. *User-programmed rules* - created from end-user programming.

3.1 Pre-programmed rules

This grouping describes systems where a developer or manufacturer fixes the rules for coordination before they are supplied to the end-users. Here professional developers try to anticipate the future use of the systems e.g. what devices will work together, what functionality the users will require etc. If there is a reliable link between the needs of the user, and that anticipated by the developer, this scheme can work well. Approaches range from systems with static functionality (that always present the same functionality to users), to systems that have some capacity to adapt to user needs. Thus, hard-coding doesn't mean that the systems are not adaptable, rather that the flexibly is constrained to pre-designed options. The defining characteristic of work contained in this category is that there is no local autonomy (the system is not able to invent its own rules but rather utilise existing rules).

Contemporary examples include task based computing systems and some forms of context awareness systems. These can adapt to the users context but are built from pre-programmed rules, switching between a pool of options according to circumstances or context, and thus can be regarded as belonging to this category.

In more detail, task based computing was pioneered by Wang and Garlan of CMU [Wang & Garlan 2000] and Fujitsu [Masuoka et-al 2003]. In this users interact with networked enabled services in terms of commanding high-level tasks (e.g. watch a movie on the largest available display). Task computing can be described as a method to enable users to discover, combine and execute coordinated contextual actions or tasks [O'Neill & Johnson 2004]. Such tasks can be regarded as composites of lower-level actions, for example the task "watch a movie on the largest available display" could be decomposed into a series of smaller steps, which would need to be combined to carry out this task. A Task Computing interface generally presents the

user with a fixed set of pre-specified tasks which, when activated, results in the system identifying and instantiating the best match of aggregated services to satisfy the requested task. Special tools such as the GUI based 'Semantic Task Execution Editor (STEER)" developed by Fujitsu have been used by developers to pre-programme tasks. A disadvantage of pre-programming tasks is that, to some extent, it involves developers guessing at the users needs. As will be described in section 4.2 the Pervasive interactive Programming (PiP) tool developed by Chin provides an alternative approach allowing end-users, rather than developers, to compose tasks.

Context-aware computing which was introduced by Schilit et al, in 1994 [Shilit 94]. There are numerous definitions of what context aware systems are, many based on the type of interactivity the system uses. Here we adopt the definition proposed by Chen and Kotz of active and passive context-awareness [Chen et-al 00]. Active context-awareness describes applications that, on the basis of sensor data, change their interface or content autonomously, whereas passive context-aware systems prompt the user to make the changes. A simple illustration of active context awareness might be a mobile phone automatically switching to silent on entering a library whereas, if it prompted the user to switch to silent, it would be an example of passive context awareness. Other examples of active context aware computing are Georgia Tech's *The Aware Home* [Kidd 99] and Microsoft's EasyLiving project in which systems adapt by switching between pre-programmed routines or states [Brumitt et-al 00]. In principle, it would be possible to employ deliberative agents to learn and evolve new rules to manage the context aware system but, currently, most systems are relatively simple and are built using pre-specified operational rules.

3.2 Agent-programmed rules

This group describes systems where the rules for coordination are formed from the use of intelligent agents, artificial intelligence or machine learning mechanisms. In general these are pre-emptive decision making systems that utlise models derived from monitoring past behaviour to predict the users next action. A distinguishing aspect of work in this group is that the agents are self-governing, by which we mean they learn new rules based on past behaviour. The performance of these systems is dependent upon there being a good match between past actions and future wants and needs

The general focus of research in this area is to create agents that can model and predict a person's behaviour, with less attention being paid to how the agents are interconnected or grouped into functional sets. Noteable test-beds and concepts for of such system have been built, such as the MavHome [Cook et-al 03], Neural Home [Mozer 98], Hive [Minar 99] and the iSpace (described later in this chapter).

The issue of how to aggregate services or devices, in the form of agents, remains a particularly difficult problem and a number of approaches have been proposed to solve this. One approach concerns the semantic web [Berners-Lee 01; Luck, 03] in which devices and their services are tagged with attributes and semantic descriptions that allow similarity searches and service aggregation to be accomplished. Another project, ANS [McCann, 04], takes the form of smart middleware which uses the OWL ontology to find appropriate devices in addition to facilitating an adaptive agent-like capability which can learn the user's preferences. This ability is also used to make decisions on device replacement. This use of OWL is somewhat similar to Chin's dComp work, described later in this paper, The multi-

agent community are also deeply emerged in the control of smart environments [Cook 06] including intelligently modelling and managing associations [McCann 04; Dulay 05]. This work utilises task-specific predefined policies to enable the agents to associate with relevant devices in their search space providing some adaptation capability to deal with devices joining or leaving the network, and people's mobility. The networking community are also undertaking research to endow network components with a degree of autonomy, generally aimed at improving throughput and reliability [Babaoglu 03, Dulay 05]. Most of these systems rely on semantic descriptions in some form. A notable exception is the work of Duman [Duman et-al 07] who has developed a function/semantic-free exploration algorithm based on fuzzy logic to discover and model the most relevant associations between devices and services operating in digital home. Finally, there are a number of wider issues that are involved in the successful deployment of autonomous agent based systems. For example all automatically constructed systems of coordinating distributed agents are vulnerable to cyclic instabilities [Zamudio et-al 08]. In addition, the risk to personal privacy due to continual monitoring is also an issue for some people [Callaghan et-al 08]. Collectively, all the above issues form a fertile and challenging research ground for researchers.

3.3 User-programmed rules

This group describes systems where the rules of coordination are formed via explicit guidance from the end-user. This approaches bypasses the manufacturer or professional developer by allowing the end user to create the system functionality directly. This methodology is commonly referred to as *end-user programming* and is characterized by the use of techniques that allow non-technical people to create programs [Cypher et-al 93].

Examples of end-user programming approaches include Humble [Humble et-al 03], which uses a jigsaw, metaphor, enabling users to snap together puzzle-like graphical programming representations as a way of building applications. The HYP system [Barkhuus 03] facilitates users to create applications for context-aware homes using a mobile phone based graphical interface. Media Cubes [Hague et-al 03] provides a tangible interface for programming an environment in which each face of a cube represents a programming operation. Programming is achieved by turning the appropriate face of the cube towards the target device. Truong's CAMP project [Truong et-al 04] uses a fridge magnet metaphor together with a pseudo-natural language interface to enable non-technical people to realize context-aware ubiquitous applications in their homes. Programming-by-example (PBE) aims to simplify the process of programming by replacing the use of programming abstractions by the use of examples that are conveyed by the user demonstrating the actions required to the system [Smith77]. Later Henry Lieberman described PBE as "*a software agent that records the interactions between the user and a conventional direct-manipulation interface and writes a program that corresponds to the user's actions*", where "*the agent can then generalise the program so that it can work in other simulations similar to, but not necessarily exactly the same as, the example on which it is taught*" [Lieberman 01]. Hence, PBE can be viewed as an approach that reduces the gap between the user requirements and the delivered program functionality by merging the two tasks. To date, the main area of PBE work has focused on graphical user interfaces running on single PCs. For instance PBE has been applied to computer application development [Myers 90], [Halbert93]; Computer-Aided Design (CAD) system [PBDCAD]; children's programs [Stagecast], [AgentSheets] [ToonTalk] and World Wide Web related technologies [Sugiura98], [Bauer 00], [McDaniel 01], [Lieberman 99], [Blackwell 01]. The MIT Alfred project and the Essex University Pervasive interactive Programming (PiP) project have extended the principles of PBE into the area of pervasive computing and Digital Homes.

The Alfred project, developed by MIT in 2002, employed the concepts 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 that 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 [Gajos et-al 02]. Whilst macros are widely used, their dependence on strict order can make them fragile and susceptible to failure, especially in applications where order of events is irregular or unpredictable. PiP, described later in this chapter, avoided this problem by using sequence-independent behaviour descriptions known as Meta-Appliances (MAps), which are akin to non-terminating processes. PiP utilises an event-based architecture and functions by mimicking examples of the required behaviour.

3.4 Supporting studies

Some insight into users wishes has been provided by a number of significant studies into 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 [Chung et-al 03]. Their findings included the need for harmonious cooperation between appliances and ease of use. However, more importantly for the work being reported in this chapter, a particularly important requirement that they discovered was the need for people to be able to customize the functionality of smart-homes. The same need was reported in a 3-year study on Finnish people undertaken by Tampere University Hypermedia laboratory [Mayra et-al 06].

3.5 Discussion

Rules underpin the operation numerous smart homes and intelligent environments. As such, rule formation provides a useful taxonomy for describing the differing approaches to programming the coordination behaviour of services available in digital homes. We have argued that pre-programmed approaches can perform well, where the user's needs and the system components can be anticipated correctly, and remain relatively static. However, in more dynamic environments, where system components and people's needs can change quickly, they are less suitable and adaptable agent based programming approaches become attractive. In these approaches, the use of automated rule generation offers the advantage of reducing the cognitive load on people by undertaking all the configuration and programming on behalf of the user. However these approaches have some drawbacks as some researchers contend that there can never be a perfect match between past actions and future needs, as users will never behave in exactly the same way from day to day, month to month and year to year. Because of this it is argued that the gap between user expectations and agent actions will remain a source of frustration, requiring agent based programming systems to be overridden by users frequently and thus annoying them. In addition, the continual monitoring of people in their homes, that agent programming approaches employ to maintain effective predictive behaviour models, is a cause of concern to anyone who worries about privacy. Finally, two significant studies have discovered there is a need for people to be able to customize their smart home functionality; an important finding that motivates further the work described in this chapter.

4.0 Pervasive interactive Programming (PiP) – An Example of End-User Programming

The users of an appliance, or the occupants of an environment, usually know the behaviour they would like from a system but do not always have the means to communicate their wishes to the system. End-user programming approaches have the advantage of not requiring the system to guess a person's wishes as people are able to explicitly describe their needs, although this is achieved at the cost of increased effort on the part of the user. In addition end-user programming goes some way to countering the fears that some people have about privacy and trust by providing more transparent operation and more user control [Callaghan et-al 08]. Thus the aim of the work described in this chapter is to develop systems that maximise user's control and operational transparency, engendering a sense of trust, and enabling people to customise the functionality of their virtual appliances and digital-homes, without requiring any detailed technical knowledge, thereby empowering user creativity. In the following sections we will explain how we have achieved this by describing how an end-user programming paradigm, programming-by-example, originally developed for "desktop computer environment", was applied to the distributed computing systems that underpin intelligent environments. We call this approach Pervasive interactive Programming (PiP).

Pervasive **i**nteractive **P**rogramming (PiP), is an end-user programming paradigm proposed and developed by Chin in 2003 as a means of addressing the issues of privacy and creativity in Digital Homes. PiP can be viewed as a methodology for enabling non-technical people to compose and orchestrate the behaviour of collections of network services, or devices, so as to produce the desired collective functionality that characterises the smart home or a virtual appliance. It provides a platform that utilises the physical user environment, with appropriate GUI support, to create a novel programming environment which enables people to customise the functionality of their virtual appliances or digital homes to suit their individual needs. Thus, a resident of a digital home, who has no technical expertise, is able to customise the functionality of coordinating network devices using natural physical interaction to demonstrate the required functionality, a task that could previously only be achieved by conventional computer programming.

4.1 PiP Terminology

A fundamental concept underpinning PiP is the notion of deconstruction and reconstruction of appliances and applications. This gives rise to the following terms:

- **Virtual Appliances** - Typical home appliances are made up from numerous services e.g. a TV is composed of a video display, audio transducer, media gateway, control interface etc. Thus, when appliances are connected to a network it might become possible to access these sub-functions, effectively deconstructing the appliance by making these basic services available to other network users. These users may in turn combine them with sub-functions from other networked appliances to form novel composite services or, as we chose to describe them *virtual appliances.* This model changes the understanding of what an appliance is, bringing with it the potential to disrupt the current appliance market by creating new types of producers and consumers. This concept is not limited to physical appliances but can, potentially, include any network service. Thus, we refer to such communities or virtual appliances by the more generic name of **M**eta **Ap**pliance**s/Ap**plication**s (**MAps**)** and the conceptual approach as the *deconstructed appliance model.*

- **Atomic & Nuclear Functions** – The virtual appliance principle depends critically on the ability to deconstruct monolithic appliance functionality, referred to as nuclear functions, into sub-functions which are offered to the network, referred to as atomic functions.

- **Meta-Appliances/Applications (MAps) –** A MAp is a description of a virtual appliance or application. MAps can be viewed as soft-objects that define the membership and behaviour of a collection of services that constitute a MAp (or virtual appliance). MAps can be designed, owned, copied, carried or traded. From a logical perspective, a MAp contains some basic properties and a collection of rules that determine the behaviour of a community of coordinating services, Rules are essentially a marriage of two different types of actions, namely antecedent (condition) and consequent (action). 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. A UML representation of MAp relationships with rules is shown in Figure 2. From the end-users' viewpoint a MAp is just a behaviour description that would create the sort of virtual appliance or environmental functionality they want. As individual end users, owners of virtual appliances or applications, have their unique preferences and their particular needs, it makes sense to let the users define theirs own MAp i.e. their own virtual appliance behaviour. MAps are created under the directions of end users to provide the functionalities they desire. MAps are akin to non-terminating processes and require no specific user expertise for their formation. They can be created, stored, retrieved, shared, executed, or removed on demand. Until a MAp is terminated, it will retain the behaviour that the user originally created i.e. it is a continually running process.

The following simple scenario is offered to illustrate the operation of MAps:

> *Janet 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 podcast. He starts up the TV application, which uses the same audio channel as his podcast, for output and tunes in to the news item. Once the news item has finished John closes the TV application and continues with his work. The podcast resumes (from the point it was interrupted) o.*

In the above scenario Johns's MAp consists of three generic virtual devices, they are: a TV device, a radio device and a podcast device, along with three virtual services: TV-control-service, radio-control-service, and podcast-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 podcast system halts its current operation and switches to play the TV audio channel" are the actions that need to be performed if the conditions are met; in this case there is only one condition. A partial definition for John's MAp is shown in Figure 11 (in the dComp ontology section).

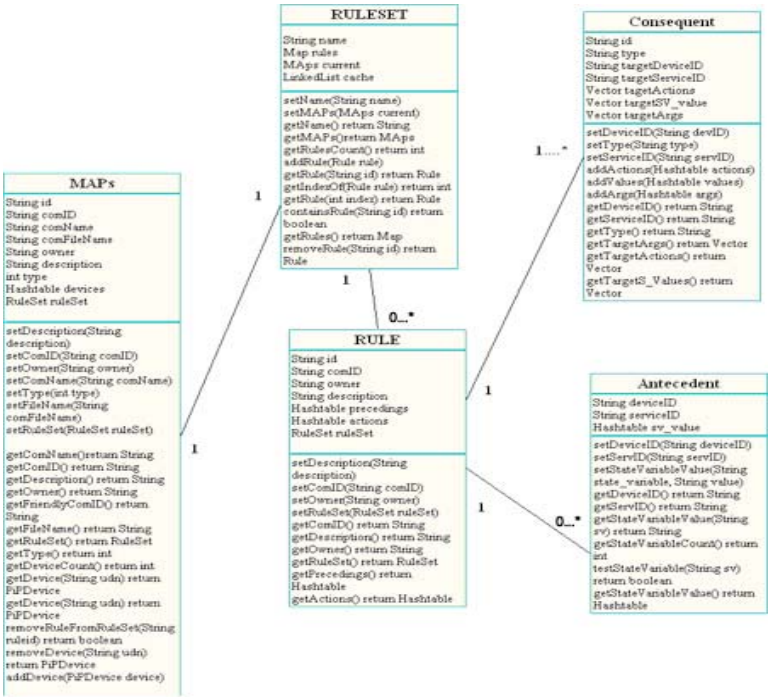

Figure 2 - The UML representation of the MAp object structure and its Rules

Having described the nature of a MAp, it may be useful to understand the differences between a task and a MAp. 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, it is an on-going (non-terminating) process that requires no specific expertise for its formation. Until a MAp is terminated, it will always provide the same functionalities that the user originally created i.e. it is a continually running process. Thus MAps are designed by users to create virtual entities that, for example, provide functionalities to customise a digital home.

4.3 PiP System Architecture

PiP was designed to work in real time within a digital home environment. It uses an event-based architecture, currently based around UPnP, to facilitate communication between components. It is based on a modular framework comprising the following six core components (see Figure 3):

1. **Interaction Execution Engine (IEE)** – this module manages communication between PiP components. It is responsible for device discovery, service events subscription, and performing network actions requests. It is built around a UPnP control point and interfaces to the low-level network layer via UPnP protocols. It features a 2-way function that, for in-bound functions, passes networked events to the Knowledge Engine and, for the out-bound functions, passes network actions, together with requests, to the network. This module also maintains and manages an event-subscription list, which it uses to store details of the devices and services that are utilised 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, triggered by the user's interactions (eg from "PiPView" or networked devices).
2. **Event Handler (EH)** – this module manages the passing of events between interested parties. Such parties need to register with the Event Handler in order to receive appropriate events and their data. Examples of events include low-level network events (e.g. device discovery), device

service events (e.g. service state changes) and high-level events caused by user interaction. Table 1 illustrates the types of events and the data handled by the EH.

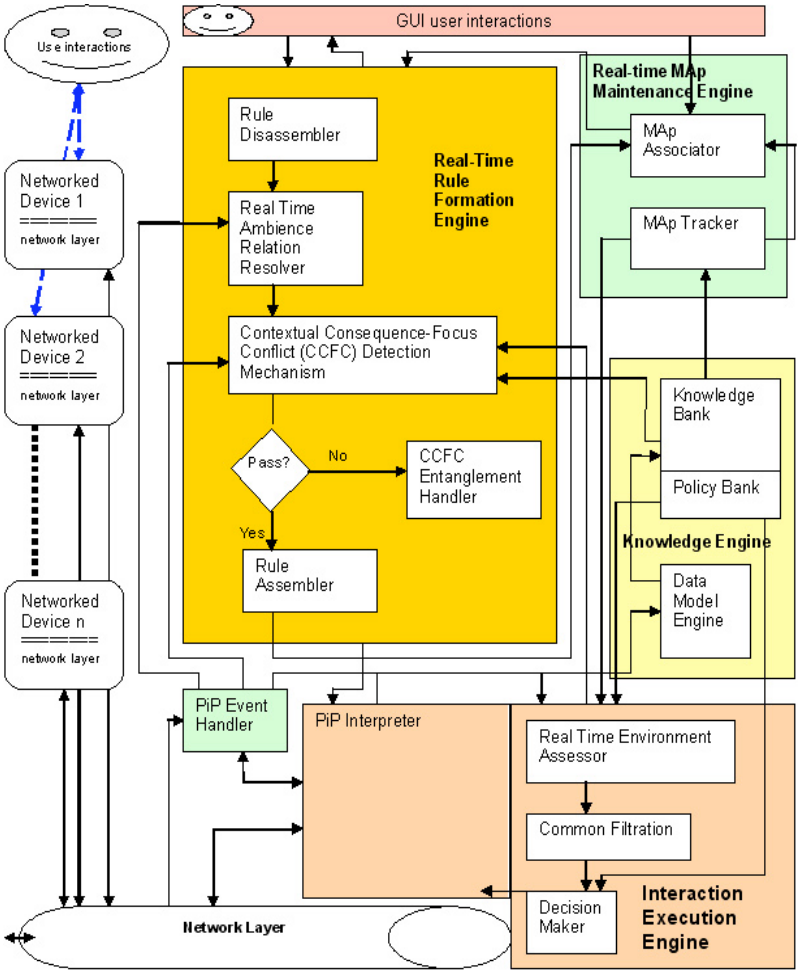

Figure 3 - PiP Architecture

3. **Knowledge Engine (KE)** – this module manages, assembles and instantiates a MAp. It also updates the record of device status as well maintaining the content of the knowledge bank and keeping it up-to-date. It notifies the Event handler when it should send a "DataModel Event" (see Table 1) to interested parties as a result of changes to the Knowledge Bank.
4. **Real-time MAp Maintenance Engine (RTMM)** — this module maintains records of all MAps. The MAp Associator registers with the Event Handler in order to receive GUI sourced user interaction action events. For example, when the user "drags & drops" devices into a "programming formation palette", the GUI component notifies the Event Handler which, in turn, generates a related event that is passed to the MAp component. Likewise, the MAp Associator Component notifies the Event Handler of any changes in the user's MAp, which, in turn, generates a MAp Event which is propagated to interested parties.
5. **Real-Time Rule Formation Engine (RTRF)** – this module assembles rules formed by monitoring user interactions during the "demonstration" mode (started and stopped by the user clicking the "ShowMe" and "Done!" button, respectively, on the PiP editor). It registers with the Event Handler to get events from the Knowledge Engine, MAp, GUI and user's activities. This module also assembles rule fragments based on user actions. At the end of a 'demonstration mode" the aggregation of rule fragments become a MAp.
6. **GUI (PiPView)** – This module provides a means for the user to interact with the system; It enables the PiP user to inspect the network environment, view online devices and services, control the physical devices and compose/delete Maps/Rules (an alternative to the use of physical devices for programming), To enhance the intuitive nature of PiP, the editor was

designed to convey the familiar "look and feel" of current 'Windows' applications. Thus the GUI utilises "drag, drop & clicking" for interaction. The software is built around a multi-threaded scheme with the 'editor class' being the main class and the user's entry point to the system. Other classes include 'pop-up dialogs', 'device control panel' and 'help'. PiPview differs to other modules in that it is not contained in every instance of PiP, rather only in the editor device (a tablet, in the current implementation).

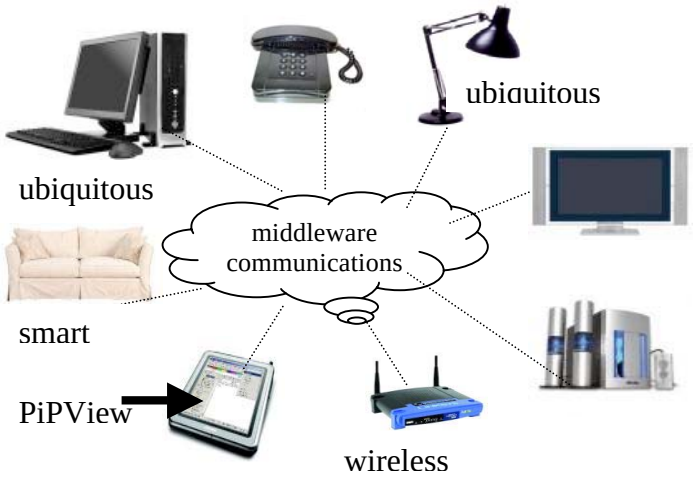

Figure 4. The PiP Evaluation Environment

4.4 HOW THE SYSTEM WORKS?

A PiP user can configure and program the functionality of a MAp using either graphical or direct physical interaction with the appliances. In PiP all networked items that exist in the ubiquitous environment can be regarded as user interfaces, as users can chose to interact with them as an alternative a GUI during the demonstration process. The network appliances used in our evaluation are shown in Figure 4. The benefit of interacting with real appliances, rather than a GUI, is that it is more natural and intuitive Thus the PiP metaphor for configuring digital home functionality is very simple.

In more detail, to create a MAp, a person needs to log into the system through PiPView. An instance of the Interaction Execution Engine (IEE) module then completes a network discovery cycle and reports available devices to the Knowledge Engine (KE) which, via the EventHandler (EH), informs all registered devices (registered at start-up time). For example, PiPView receives a notification and then an instance of the interpreter transforms and renders descriptions of the devices on the GUI, allowing the user to decide which devices to use for creating or modifying a MAp.

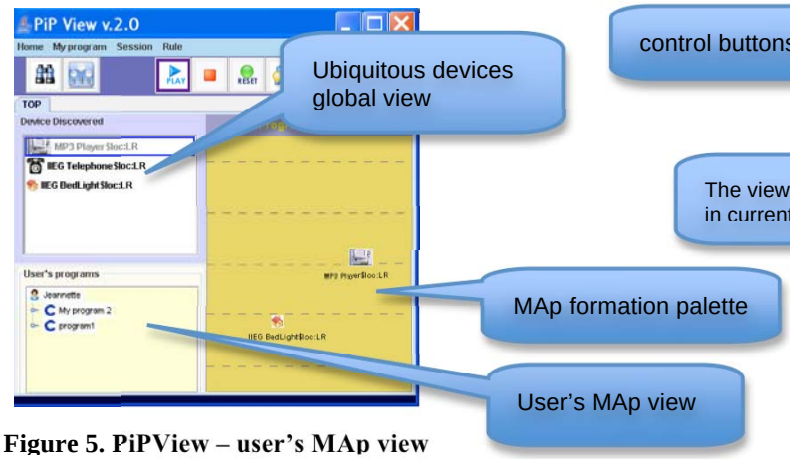

Figure 5. PiPView – user's MAp view

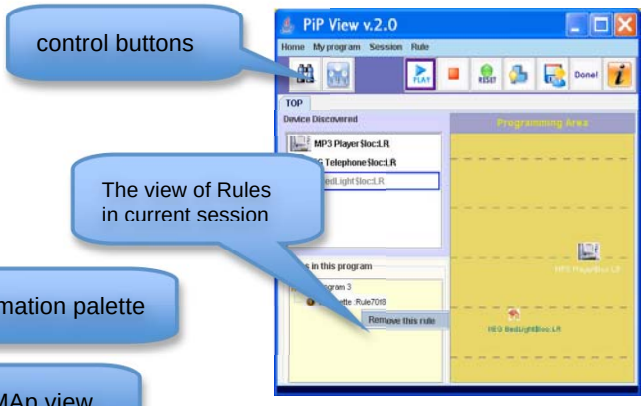

Figure 6. PiPView – Rules view

MAps are created by the user, who first assembles a community by dragging and dropping service and device representations into a formation palette. This defines the set of devices that, via the Event Handler and the Interaction Execution Engine, share events and become a virtual appliance or MAp. When devices are formed into assemblies the user is free to save or remove the MAp at any time during the demonstration cycle.

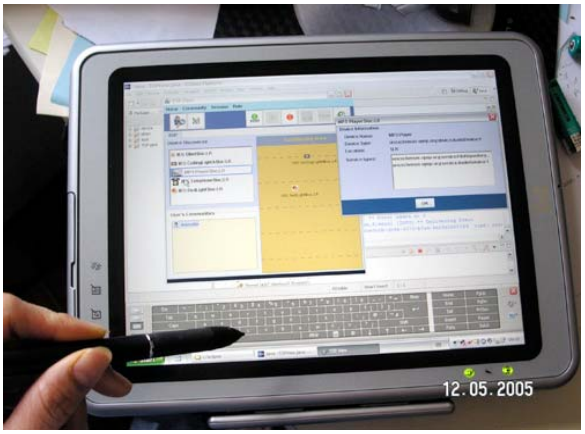

Figure 7. PiP on tablet view

Figure 8. End-User programming via physical interaction

Using PiPview the user informs PiP when to the process of demonstrating the MAp or virtual appliance behaviour will begin which, in turn, activates the Real Time Rule Formation (RTRF) Engine. Next the user demonstrates actions as to how the MAp should behave by providing examples using any one of three methods:

(1) Physically interacting with the devices themselves;
(2) Using the GUI;
(3) A combination of both.

Using the users preferred interaction method, the user demonstrates the desired behaviour which results in the generation of related events on the network. In the background the Real-Time Rule Formation Engine "listens" for user's activities, which are communicated to it via the Event Handler that in turn, is informed by the Knowledge Engine when it receives a notification that the status of a remote device has changed. This behaviour is then encoded as a set of rules as described earlier.

In terms of sequencing, PiP aims to give the user as much freedom as possible, allowing antecedents and consequents to be formed in any quantity and order (i.e. the user is not required to follow a rigid logical sequence). Thus, unlike macro languages, where the sequence of instructions or actions is important, PiP places no significance on logical sequence. PiP employs a rule policy to maintain a MAp execution process in which "a set of conditions is satisfied if, the conditions defined within the context of this set, are all satisfied". For example the rule 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 to any of the following rule statements:

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

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

This is achieved by using a Rule Disassembler which separates antecedents and consequents. The rules are then shaped by a Real-Time Ambience Relation Resolver, which is responsible for resolving the relationships of the information received from the network‘s internal system and the user. To avoid rule conflicts (eg contradictory condition-actions) a process, the Contextual Consequence-Focus Conflict (CCFC) Detection Mechanism, tests whether the user’s current action conflict with other MAps rules which may result in unwanted system behaviours. If the CCFC detects no conflicts then the Rule Assembler constructs the rule. If there are conflicts, the CCFC Entanglement Handler will deal with this situation by performing the following operations (1) gathering the conflict information (2) isolating the conflict actions from those in the current MAp and (3) presenting conflicts to the user so he may alter his action to avoid the conflict. Having created a MAP, to activate it, the user simply needs to drag the MAp’s 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. The dComp Ontology

To maximise the usability of the system, data needs to be formed in a suitably semantic and standardised manner so that it can be understood and reasoned on by all parties in the network. To support this we have devised an ontology for PiP called dComp, that supports information interoperability between applications and provide a common knowledge framework. PiP leverages the dComp ontology semantics as a core vocabulary for its information space. dComp allows information to be conceptually specified in an explicit way by providing definitions associated with names of entities in a universe of discourse e.g. classes, relations,

functions, or other objects, that are both in a machine and human useable format. In more practical terms, the dComp ontology describes, for instance, what a device names mean, and provides formal axioms that constrain the form and interpretation of terms.

DCOMPDevice Class	**DCOMPHardware Class**	**DCOMPService Class**	**Rule Class**	**Policy Class**
DCOMPDevice	Hardware	DCOMPService	Rule	Policy
MobileDevice	CPU	LightsNFittingsService	UnchangableRule	Mode
StaticDevice	Memory	LightService	PersistentRule	**Time Class**
NomadicDevice	DisplayOutput	SwitchService	NonPersistentRule	**DCOMPperson Class**
Light	DisplayScreenProperty	TelephoneService	Preceding	
Switch	AudioOutput	AlarmService	Device	
Telephone	AudioOutputProperty	TemperatureService	**Preference Class**	**Action Class**
Alarm	Tuner	EntertainmentService	Preference	Action
Blind	Amplifier	AudioService	SituationalCondition	PermittedAction
Heater	**DCOMPCommunity Class**	VideoService	CommunityPreference	ForbiddenAction
FileRepository	SoloCommunity	FollowMeService		Recipient
DisplayDevice	NotJointCommunity	SetTopBoxService		TargetAction
AudioDevice	PersistentCommunity	StateVariable		
SetTopBox	TransitoryCommunity	TOPService		
Characteristic	CommunityDevice			
DeviceInfo				

Table 2- dComp ontology

5.1 dComp Rationale

The dComp ontology was built on the OWL language, as it is more expressive than RDF or RDF-S that is it provides additional formal semantic vocabularies allowing PiP to embed more information into the ontology. In addition OWL is widely used, especially for the semantic Web, with numerous supporting tools such as Jena [HP Jena] and inference engines such as RACER [Haarslev and Moller 01], F-OWL [Zou 04], and Construct [Network Inference]. In order to realise our vision, a set of explicitly well-defined vocabularies (i.e. an ontology) was needed to model, not just the basic concept of decomposed devices but also, the communities they form, the services they provide, the rules and policies they follow, the resultant actions that they take, and of course the people who inhabit the environment along with their individual preferences; dComp provides these properties. In creating dComp we have sought, wherever possible, to build on existing work. One such notable ontology is SOUPA [SOUPA] from Ubicomp which, while aimed at pervasive computing, lacks support for key PiP mechanisms such as community, decomposed functions and coordinating actions needed to produce higher level meta functionality [Soupa]. In addition, at the time dComp was developed the SOUPA standard had only limited support for the concept of UPnP-based devices which PiP depends upon. However, SOUPA has a well-defined method of supporting notions of Action, Person, Policy and Time which dComp adopted. Thus, most of the innovation in dComp relates to the ontology of decomposition and community leading to the name "**D**ecomposed **Com**munity **P**rogramming" (dComp).

5.2 The dComp Ontology – An Overview

The following description takes the form of a summarised walk-through dComp; the full specification is available online [DCOMP]

5.2.1 The Device Class

The main class is called "DCOMPDevice" and provides a generic description of all PiP devices. Currently DCOMPDevice has 10 sub-classes (see table 2) including both nuclear (traditional appliances) and atomic (decomposed) devices and remains the subject of ongoing development. The roles of most sub-classes are obvious from their names. Those which might not be obvious include "DeviceInfo" which is for individuals that share some UPnP descriptions, "DeviceInfo" is used for devices sharing some UPnP descriptions, "Characteristic" for different mobility characteristics, Relationships are defined by using the OWL object property and are: (1) hasDeviceInfo (2) hasHardwareProperty (3) hasDCOMPService (4) hasCharacteristic. The main elements of a typical DCOMPDevice expression is shown in figure 9.

5.2.2 Hardware Class

An abstract class, DCOMPHardware, generalises all PiP hardware that exists in a DCOMPDevice and, in the current version, has 8 sub-classes along with associated properties: CPU, Memory, DisplayOutput, DisplayInput, AudioOutput, AudioInput, Amplifier and Tuner. In order for the PiP DCOMPDevices to work together, every DCOMPDevice on the dComp network, offers services. These services are modelled by a class called "DCOMPService" which currently contains three sub-classes, namely PiPService, LightsAndFittingsService and EntertainmentService. Each contains sub-services, for example, the EntertainmentService class includes AudioService, VideoService, FileRepositoryService, SetTopBoxService and FollowMeService. The LightsAndFittingsService and EntertainmentService are mutually distinct (ie in mathematical terms, they do not belong to a same set). These characteristics are modelled by declaring the classes to be disjointWith each other. Every service in the dComp environment is identified by a property called "serviceID" and a class called "StateVariable" (to represent UPnP values). The StateVariable class has three properties, namely: "name", "value" and "evented". The relationship between a DCOMPService and the StateVariable is linked by an object property called "hasStateVariable". The relationship between a DCOMPDevice and DCOMPService is coupled by an object property called: hasDCOMPService.

```
<device:AudioDevice rdf:ID="TestDevice12">
<device:hasDeviceInfo>
<device:DeviceInfo>
<device:friendlyName>TestDevice12</device:friendlyName>
<device:DeviceUUID>0</device:DeviceUUID>
<device:DeviceType>urn:schemas-upnp-
org:TestDevice12:1</device:DeviceType>
<device:DeviceModelURL>http://TestDevice12URL/</device:Devi
ceModelURL>
<device:DeviceModelNumber
rdf:datatype="&xsd;double">0.0</device:DeviceModelNumber>
</device:DeviceInfo>
</device:hasDeviceInfo>
<hw:componentOf>
<hw:RAM rdf:about="#JCTestMemory2"/>
</hw:componentOf>
<serv:hasDCOMPService>
<!-- can have more than 1 service -->
<serv:AudioService rdf:about="#JCAudioService01"/>
</serv:hasDCOMPService>
<!-- 2nd service -->
<serv:hasDCOMPService>
<serv:AudioService rdf:about="#JCAudioService02"/>
</serv:hasDCOMPService>
<!-- 3rd service -->
<serv:hasDCOMPService>
<serv:AudioService rdf:about="#JCAudioService03"/>
</serv:hasDCOMPService>
```

Figure 9 - Typical Display Device Expression

```
<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 rdf:about="#JC"/>
</com:hasOwner>
<com:hasCommunityDevice>
 <com:CommunityDevice rdf:about="#JCMonitor CRT17"/>
</com:hasCommunityDevice>
<com:hasCommunityDevice>
<com:CommunityDevice rdf:about="#JC AudioMMS223"/>
</com:hasCommunityDevice>
<com:hasCommunityDevice>
 <com:CommunityDevice rdf:about="#JC :NetGem442"/>
</com:hasCommunityDevice>
</com:TransitoryCommunity>
```

Figure 10 - Typical TV community definition

```
<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 11- John's Meta-TV Appliance

5.2.3 Community Class

In order to support the notion of community (a MAp), dComp uses a class called DCOMPCommunity. In the current implementation three types of communities are used namely: (1) SoloCommunity (for those devices not yet part of a community) (2) PersistentCommunity (for communities with a degree of permanency) (3) TransitoryCommunity (for communities with a short lifetime). A dComp device (DCOMPDevice) can join one or more communities (a community must have at least one device). Relationship between a dComp device (DCOMPDevice) and a dComp community (DCOMPCommunity), is described using an object TransitiveProperty called "inTheCommunityOf". A class called "CommunityDevice" is introduced to represent all the devices in a community. Devices are identified by an object, deviceUUID.. The relationship between a Community and a Community device (CommunityDevice) is linked by another object TransitiveProperty called "hasCommunityDevice". A user forms communities in dComp; thus, each community has an owner. The properties of Communities are: community ID, communityName, communityDescription and timestamp. The relationship between a community and its owner is linked by an object type property, called "hasOwner". An example of the main elements in a dComp TV community is given in figure 10.

5.2.4 Rules Class

PiP uses rules for coordinating community actions. These are supported by a class called "Rules" which models three types of rules: (1) UnchangeableRules (rules that can not be changed), (2) PersistentRules (rules that infrequently change) and (3) NonPersistentRules (rules that frequently change). These rules are mutually distinct and are declared to be complementOf each other. Rules have properties: ruleID and ruleDescription and an object property called "hasRuleOwner" to link to the owner (the rule and community owners may be different people). A class called "Preceding" is used to represent a set of triggers that cause

the coordinating actions to be executed. The devices in the Preceding class are identified by their deviceUUID, and the service they offer. Lastly an object property called "hasAction" binds the relationship between Rules and Actions. The main elements of a Rule Definition is given in figure 12.

```
<NonPersistentRules rdf:ID="Rule1">
 <rule:ruleID rdf:datatype="&xsd;int">00001</rule:ruleID>
  <rule:ruleDescription>Test Rule 1</rule:ruleDescription>
  <com:communityID>Tran-JCTV</com:communityID>
 <rule:hasRuleOwner>
   <person:Person rdf:about="#JC"/>
  </rule:hasRuleOwner>
<rule:hasPreceding>
  <!-- can have more than 1 device -->
  <rule:Device>
   <dComp:DeviceUUID>uuid:Telephone01</dComp:DeviceUUID>
    <serv:hasDCOMPService>
    <!-- a device can provide more than 1 service -->
     <serv:TelephoneService>
      <serv:serviceID>Telephone</serv:serviceID>
       <serv:hasStateVariable>
                 <!-- a service can have more than 1 value of state variable-->
                   <serv:name>state variable 1</serv:name>
                   <serv:value>RINGING</serv:value>
     </serv:hasStateVariable>
     </serv:TelephoneService>
    </serv:hasDCOMPService>
  </rule:Device>
</rule:hasPreceding>
<rule:hasAction>
 <act:PermittedAction rdf:about="#TestAction"/>
</rule:hasAction>
</NonPersistentRules>
```

```
<owl:Class rdf:ID="SituationalCondition">
  <rdfs:label>SituationalCondition</rdfs:label>
  </owl:Class>
  <SituationalCondition rdf:ID="DuringTheWorkdays"/>
  <SituationalCondition rdf:ID="DuringTheWeekends"/>
  <SituationalCondition rdf:ID="WhileOutOfTown"/>
  <SituationalCondition rdf:ID="WorkingFromHome"/>
  <SituationalCondition rdf:ID="FriendsVisiting"/>
  <SituationalCondition rdf:ID="FamilyVisiting"/>
  <SituationalCondition rdf:ID="OnHoliday"/>
  <SituationalCondition rdf:ID="WhenComeHomeFromWork"/>
  <SituationalCondition rdf:ID="WhenComeHomeFromSchool"/>
  <SituationalCondition rdf:ID="WhenAtMyOffice"/>
  <SituationalCondition rdf:ID="WhenDining"/>
  <SituationalCondition rdf:ID="WhenHavingLunch"/>
  <SituationalCondition rdf:ID="WhenHavingBreakfast"/>
  <SituationalCondition rdf:ID="WhenEating"/>
  <SituationalCondition rdf:ID="WhenPlayingComputerGames"/>
  <SituationalCondition rdf:ID="WhenWatchingTV"/>
  <SituationalCondition rdf:ID="AtNight"/>
  <SituationalCondition rdf:ID="InTheMorning"/>
  <SituationalCondition rdf:ID="AtLunchTime"/>
  <SituationalCondition rdf:ID="AtTeaTime"/>
  <SituationalCondition rdf:ID="Alone"/>
  <SituationalCondition rdf:ID="WhenAlarmGoesOff"/>
  <SituationalCondition rdf:ID="WhenSmokeAlarmGoesOff"/>
```

Figure 12 - Main elements of Rule Definition **Figure 13 - Main elements of a Situated Condition.**

```
<act:PermittedAction rdf:ID="TestAction">
 <act:actionName>Test action</act:actionName>
  <act:hasRecipient>
device:DeviceUUID>UUID:PHLAudioMMS223</device:DeviceUUID>
    <serv:serviceID>AudioMMS223</serv:serviceID>
  </act:hasRecipient>
 <act:hasTargetAction>
   <act:actionName>Mute</act:actionName>
   <act:targetValue>Mute</act:targetValue>
 </act:hasTargetAction>
</act:PermittedAction>
```

Figure 14 - Main elements of an Action (muting the TV)

5.2.5 Action, Person, Policy and Time Class

As mentioned earlier, wherever possible dComp builds on existing ontology work. As SOUPA provides a suitable DCOMPperson, Policy and Time ontology these have been adopted in dComp. The dComp Action ontology document has, to some extent, been influenced by the SOUPA Action ontology. The class "Action" represents the set of actions defined by the rules. As with SOUPA, dComp provides two types of actions, namely: PermittedAction and ForbiddenAction class. The Action class in dComp is the union of these two action classes; every coordinating action has its target devices. A class called "Recipient" models target devices, which represents a set of target devices where actions take place. The members of Recipient are identified by their deviceUUID and the serviceID. Actions for the recipient are called "TargetAction" which has two properties namely actionName (the name of the action) and targetValue (the value for the action to be taken). A typical statement "when the phone rings, mute the TV" could be expressed as in figure 14.

5.2.6 Preference Class

A person's preferences are described in dComp by DCOMPPreference. In dComp, preferences are referred as "situated preferences", which is similar 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 [Vastenburg 04]. The "Preference" class represents a set of situated preferences of a person for his community. This Preference class has a subclass called "CommunityPreference" and an associated property called "communityID". To model "person A prefers X, depending on the situation conditions of Y", another class called "SituatedConditions" is defined which represents the set of situated conditions that the person's preferences depended on. Although users are allowed to define their own "SituatedConditions", dComp explicitly defines a list of pre-set situated conditions so that it forms a default template that a person can use. The Preference class has a close relationship to the Person class. To bind this relationship, an object property called "hasPreference" is used, which links the domain of Person to the range of Preference. The relationship between the Preference class and SituationConditions class is linked by another object property called: "hasCondition". The main elements of a Situated Condition are given in figure 13.

6. Evaluation

PiP was designed to be used by people and thus, to evaluate it we devised a small "proof of concept" trial involving real users who were asked to compose bespoke MAps within an experimental digital home, the iSpace. The primary purpose of the evaluation was to determine if users were able use PiP in a creative manner to construct MAps of their own design and to gain an insight into the participants post-trial views of PiP's usability. In addition we tested the performance of dComp, as ontology can be computationally demanding. We summarise this work in the following sections.

.

6.1 PiP Testbed

Figure 15 – The Essex iSpace

Pip was evaluated in Essex the iSpace, a test-bed called which takes the form of a two bed roomed domestic apartment, see Figure 15. The iSpace was built from the ground-up to support digital home research and has many special structural features such as cavity walls/ceilings containing power & network outlets together with provision for internal wall based sensors and processors etc. In addition the iSpace has been populated with in excess of a quarter of a millions pounds worth of networked equipment varying from appliance, sensors, actuators through to special purpose equipment to support user trials. There are numerous networks in place ranging from wired , power-line, wireless, broadband to high-bandwidth multi-mode fibre connections to the outside world. The network and middleware infrastructure is illustrated in figure 16. All the basic services are electrically controlled wherever possible (eg heating, water, doors, telephones, MP3 players, lights, etc).

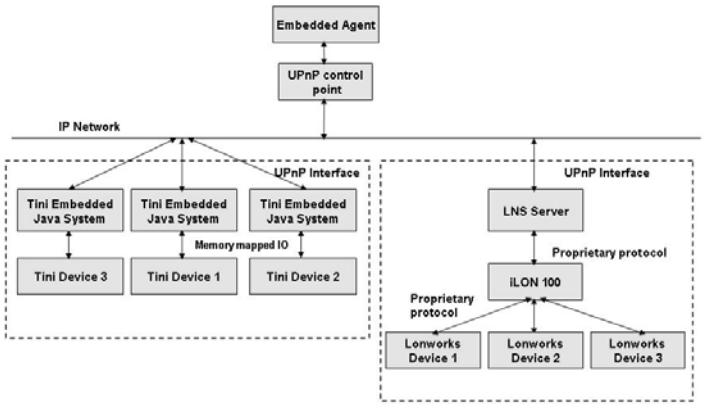

Figure 16 - The iSpace Network Structure

6.2 dComp Performance Evaluation

Ontology brings many benefits to PiP, such as the ability to employ reasoning about service selection and aggregation but its downside is that it can be computationally demanding. In our case there was concern that one of the special features of PiP, the MAps based decomposed descriptions, might not perform well in large domains because of increased link following. To evaluate the performance of dComp we compared two sets of device descriptions; the first description was structured in typical xml-based "all-in-one" format, while the second was decomposed into smaller segments (i.e. broken up into hardware and service information), each segment being "linked" back to the device. Both descriptions were written in OWL. For each set, we used 2 different quantities of devices in the test (3 and 32). A common query with five conditions was used for the test, with each test being run fifty times. The test was conducted on a WindowsXP, 2.08 GHz, 512 RAM machine. Four sets of tests were completed: (1) 3 device descriptions in "all-in-one" format (2) 3 device descriptions in "decomposed" format (3) 32 device descriptions in "all-in-one" format and (4) 32 device descriptions in "decomposed" format.

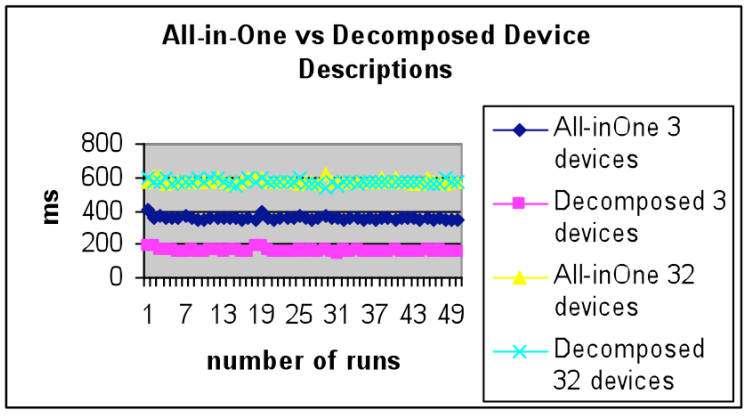

Figure 17. A typical dComp performance test

A representative example of our tests is shown in figure 17. As can be seen we found that the decomposed device description out-performed the compact devices description for smaller domains with fewer devices. On average, queries took half the time that "all-in-one" format descriptions took. Although we had been concerned that decomposed descriptions might not fare as well for larger domains we found that this was not the case, as the system performed as well as the "all-in-one" descriptions, whilst offering the

advantages of decomposition described earlier. This we attribute to additional link-processing being counterbalanced by the processing benefits of smaller, better focused descriptions. For larger domains we found that the performance of decomposed versus the compact descriptions remained roughly the same.

6.3 The PiP Evaluation

To assess the participants' subjective views on the usability of PiP, an evaluation methodology was developed with the assistance a socio-technical research unit, Chimera, based at the BT Research Labs in Suffolk, England [DiDuca et-al 05]. The evaluation comprised both observations and a questionnaire measuring attitudes over six usability dimensions shown in Table 3 (a higher rating score on the dimensions shows greater usability). The questionnaire was developed to assess the participants' subjective judgments about the usability of PiP. It consisted of a set of seventeen statements, measuring attitudes over the following six usability dimensions: the overall concept, user controls, cognitive load, information retrieval/visualisation, affective experience and future potential. The questionnaire was based on a five-point Likert scale with responses from "Strong Agree" through to "Strongly Disagree". The dimensions each consisted of a series of statements (from 2 to 4) with 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. In the research community 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. The questionnaire was piloted on 3 users and the feedback used to refine the procedures and questionnaire for the main trial. In terms of the structure of the trials, our strategy was to set-up as open an arrangement as possible, with minimal constraints being placed time, methods, and tasks 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.

6.3.1 Participants

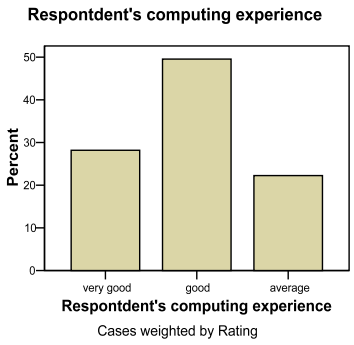

Figure 18 - Participant's Computing Experience

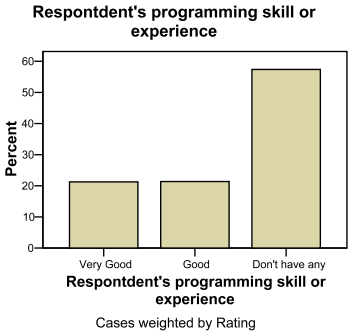

Figure 19 - Participant's Programming Experience

The PiP evaluation comprised eighteen participants drawn from a diverse set of backgrounds (e.g. housewives, students, secretaries, teachers etc), see figure 20. The gender mix was 10 females and 8 males with ages ranging from 22 to 65. The participants also formed a multicultural group including Asians, Europeans, Americans, Latin-Americans and

Australians. All trial participants had some minimal computing experience (i.e. they knew how to use a mouse and keyboard) see Figure 18. 60% of the participants had no programming experience whilst 20% of them had a very good knowledge of programming, see Figure 19. For the PiP evaluation sessions, participants were given five sets of devices drawn from a set of lights, a telephone, smart sofa and an MP3 player and asked to create a collective behaviour of their own design. In addition PiPview (the Pip GUI) was set-up to run on a winXP tablet PC (HP) that connected to the iSpace network via a Linksys 802.11g WIFI access point. As will be explain in greater detail below, the evaluation was preceded by a 20-minute training session to show the participants how to use the PiP technology. With only 5 devices, the possibilities for the users to create interesting designs were a little limited. However, despite that limitation, the users created a number of interesting virtual appliances such as, for example, Telight (telephone linked to lighting), LightSof (reading lights linked to sitting on the sofa) and Telight3 (lights and MP3 player responding to the telephone).

6.3.2 Procedures

The University enforces stringent ethical regulations pertaining to research involving people and animals. Thus, prior to the evaluation, a consent form was prepared and completed by all participants before commencing their sessions 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.

Each PiP evaluation participant was given a 20-minutes training session to familiarise them with PiP. This training session included: a briefing on the PiP concept, a walk through using the PiP UI, a quick demo on how to compose a MAp followed by an introduction to the trial environment. The task for the trials was deliberately open and participant were use PiP to customise the functionality of the 5 devices they were given in any way they wanted; thus the participants were free to create one or more MAps of their own design. After creating MAps participants were encouraged to switch between the MAps they had created.

Figure 20 – Trial Participants

No time limit was set for the participants to customise the space. Assistance was provided where needed. Following completion of the evaluations, a questionnaire was administrated to measure the

participants' subjective judgements of PiP. Participants rated a total of seventeen statements covering six dimensions mentioned above. 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. Data was analysed using SPSS.

6.4 Results

6.4.1 Performance

An analysis of the evaluation data showed that, in general, all the dimensions rated well (scoring above 4) indicating the users were generally well satisfied with the system. At the outset of the work, two suppositions that we wished to confirm were that people would enjoy the experience of using PiP to create MAps and find the process relatively easy. Both of these suppositions were supported by the evaluation results as, 'enjoying the experience' (the mean of the affective dimension) scored 4.6, the highest rating, whilst the cognitive load dimensions achieved an overall average of 4.3 indicating people found the process relatively simple. In fact, it was found that 88.9% reported that they used the controls with ease and 83% of participants were able to use the system to create their desired environments with little or no assistance.

The evaluation showed that after only a brief training session (20-minutes), 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 (although it was typically of the order 2-5 minutes, depending on the complexity of the behaviour being designed). Concerning the two methods available for demonstrating examples, 11% of the participants chose to customise their personal space wholly via GUI controls while 72% of them conducted by physical interactions with the environment while the rest used a mixture of both. Trial participants showed no sign of distress 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 remainder 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.

6.4.2 Questionnaire Rating

A variety of tests were completed to analyse the questionnaire ratings using the SPSS software package. Table 2 summarises the overall rating scale for the six dimensions evaluated. The results revealed 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. 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. Thus these results

supported the original supposition that people would both enjoy and find the process of using PiP easy.

	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 3 One-Way ANOVA test on dimension vs qRating

In addition, we completed a cross analysis based on the participants computing and programming experience ranging from average, good to very good (Figure 19). 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 21). Examples of remarks recorded from this group include: "I just feel like right now I want to sit down for a lot longer and try out all sorts of MAps that I could possibly create!" and another one : "I can really get quite keen on it".

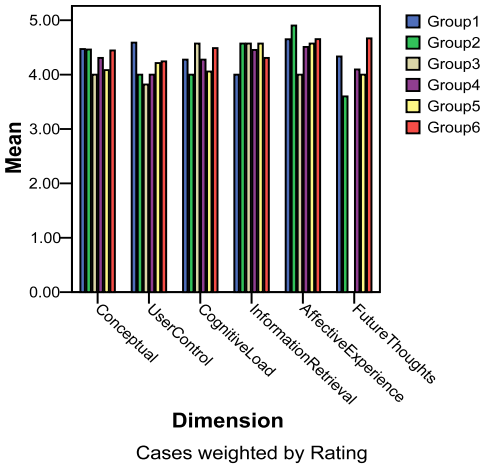

Figure 21 - Mean ratings for each individual group of participants.

In general the "Information Retrieval" dimension (how well information was conveyed to the user) scored the least indicating this was they worst aspect of the PiP design. Given that the interface was GUI was a rather crude prototype, not the final product, it was not surprising to find that the interface could be improved. Other useful findings were that, we found no significant variation across culture but found some minor variation on cognitive loading for age groups, with younger participants finding the system slightly easier to use. In terms of general observations, none of the participants appeared to find the principles difficult to understand. By way of an example, that was typical of many users, one participant stated "*I thought the basic principles themselves are very simple and straight forward*", "*I felt I could*

easily grasp the basic principles". This particular comment was from the group with no programming skills at all, a key target group for PiP. Overall 83.4% of all participants found PiP intuitive to use and 94.4% of all participants stated they felt it rewarding to use PiP. Thus these initial results support the original hypothesis of the work that it is possible to produce a system that empowers non-specialists to be able, and to enjoy, programming customised MAps.

Like all disruptive technologies, user power is significant but, ultimately, just one of many factors in determining whether a given technology will be adopted in the market. This evaluation stops short of understanding the issues that drive companies into adopting particular products although, for those interested in understanding those processes an illuminating paper has been written by a member of Intel's User Experience Group which takes PiP and the concept of MAps as its focus (Johnson et-al 08).

7.0 Concluding Discussion

In this chapter we have described a vision for customising intelligent environments based on a form of programming-by-example. In this approach non-technical occupants of intelligent environment are given end-user tools to enable them to create their own bespoke functionalities for networked environments. In order to achieve this vision we introduced a number of new concepts and methodologies, in particular the 'deconstructed appliance model', meta-appliances/applications (MAps), the dComp ontology and Pervasive interactive Programming (PiP).

In order to situate our work we presented a *'rule formation'* taxonomy based on the way rules in networked devices are constructed; Pre-programmed rules - Agent-programmed rules - User-programmed rules. The work in this chapter lies firmly within the latter class, user-programmed rule based systems. This allowed us to contrast our work to other significant approaches, such as autonomous self-learning agents. The ensuing review revealed that all approaches have strengths and weaknesses meaning that the choice of approach depends on the specific needs of a given applications. For instance self-programming agents do extremely well where there is a need to reduce the cognitive load on an individual, as they can manage the whole process without intervention from the user. However, for cases where there is a need for the home occupant to exercise greater control, or to participate intimately in the creative design process, then end-user programming has advantages. Of particular relevance to this chapter is "programming-by-example", a well established and successful end-user programming paradigm. We described programming-by-example from its roots in the mid-seventies when Smith introduced it through to the inspirational work of Lieberman in the 90's and to its use by Chin in pervasive computing environments in the new millennium. Over this time, programming-by-example has evolved from a means of programming applications on single platforms, to programming behaviours of embedded systems in distributed computing platforms. As part of this review we reported on a number of significant studies into smart home requirements and reported on their finding that a particularly important requirement they found was the need for people to be able to customize the functionality of smart-homes. This finding underpins the motivation behind this work.

By way of an example of end-user programming we presented Chin's work on Pervasive interactive Programming (PiP). Chin's work is novel in that it translated the principles of

programming-by-example from single stand-alone computing platforms to distributed computing platforms. Before PiP, programming-by-example had not been applied to programming tangible physical objects, especially distributed embedded computing nor any other aspect of pervasive computing. PiP also introduced a number of innovative concepts such as the deconstructed appliance model, virtual appliances and Meta-Appliances/Applications (MAps). These concepts represent a radically new way to view the nature of home appliances by breaking apart monolithic appliance functionality into their elemental or atomic components, allowing them to be recombined by the user so as to customise the functionality of their appliances or environment. Currently, for historical reasons (ease of use, economies of production, technology limitations etc), appliances have been largely monolithic units (e.g. televisions, telephones etc), a paradigm that 'deconstruction' challenges. Whilst such changes to the market with replacement of monolithic appliances by more elemental services in the home would require a somewhat abrupt change to markets and therefore may be unlikely a more gradual evolution is possible by the gentle augmentation of appliances with network capabilities. Thus, even if it came to be that future homes had elemental services installed as standard for example displays, audio transducers, sensors, actuators, tuners, streamers etc, much as heating and lighting services are now standard, the road to that future is more likely to be a gradual evolution. This chapter also revealed another important feature of MAps; they are soft-objects and portable, able to move with people and, where possible, configure the environment to reflect a person's preference wherever they are. In addition, beyond the home, MAps have the potential to alter business models as, being soft-object created by people with no technical skills, but having value, they could be traded. The soft nature of MAp makes them ideal for online web based trading which could be undertaken could be by individuals, new types of business coalitions, or traditional companies.

We described our prototype 'proof of concept' architectural implementation of PiP. The core principles included the use of eventing, to capture user interaction; virtual engines, to execute user generated PiP rules and ontology, to allow better resource sharing and allocation. We provided an overview of the dComp ontology, which built on both Soupa and Owl principles to provide support for the PiP deconstructed model, especially MAps. In particular dComp differs from other ontologies by providing representations for community, decomposed functions and coordinating actions which are fundamental to the PiP model for end-user customization.

The PiP evaluation was conducted in a purpose-built intelligent environment testbed known as the iSpace. This is a purpose built domestic apartment, which contains numerous, networked appliances, sensors and actuators. Initial testing of the dComp ontology showed that even the most computationally intensive queries relating to PiP decomposition were returned in less than a second, which yielded acceptable system performance. The main evaluation concerned assessing the ease or difficulty people had in using our prototype system PiP system to programme MAps. Whilst we were only been able to undertake a comparatively small scale evaluation with 18 users, the initial findings were most encouraging as they showed that it was possible to produce an end-user programming system that empowers non-specialists to be able, and to enjoy, programming coordinated actions of distributed embedded computer systems in a digital home. Remarks such as "*I felt I could easily grasp the basic principles*", from participants with no programming skills at all were particularly encouraging as such people were a key target of our end-user programming of digital homes work.

Finally, this chapter has set out to explain what end-user programming is, how it can be implemented and the benefits it can offer occupants of future high-tech homes. Whilst this chapter has argued strongly in favour of end-user programming and, especially, programming-by-example the future is rarely so 'black and white' and it is likely that solutions will be hybrids of many ideas. However, we feel passionately that whatever final solutions emerge, they should to recognise the human condition by addressing the fundamental needs of people to be creative and to protect their privacy.

Acknowledgements:
We wish to thank Martin Colley (Essex University), Hani Hagras (Essex University), Malcolm Lear (Essex University), Phil Bull (BT), Rowan Limb (BT), Brian Johnson (Intel) for their role in encouraging and motivating this work in various ways, not least by their stimulating discussions and papers. In addition we are pleased to acknowledge the DTI (Next Wave Technologies and Markets programme) who funded the original phase of the work, Chimera (Institute for Socio-Technical Research) who advised on the evaluation and, finally, Essex University who funded the PhD scholarship.

References:

[Augusto 06] Augusto, Juan Carlos; Nugent, Chris D. (Eds.) "*Designing Smart Homes: The Role of Artificial Intelligence*", Lecture Notes in Computer Science , Vol. 4008, 2006, ISBN: 978-3-540-35994-4

[Babaoglu 03] O. Babaoglu et al., "*Anthill: A Framework for the Development of Agent-based Peer-to-Peer Systems*", 22nd International Conference on Distributed Computing Systems, 2003.

[Bauer 00] Mathias Bauer, Dietmar Dengler, Gabriele Paul, Markus Meyer, "*Programming by example: programming by demonstration for information agents*", Communications of the ACM, Volume 43, Issue 3 (March 2000), pp.98 – 103

[Berners-Lee 01] Berners-Lee T, Hendler J, Lassila O. "*The Semantic Web*", *Scientific American*, May 2001

[Barkhuus 03] Barkhuus, L., Vallgårda, A: "*Smart Home in Your Pocket*", Adjunct Proceedings of UbiComp 2003 (2003) 165-166

[Blackwell 01] Alan F. Blackwell, Rob Hague, "*Designing a Program Language for Home Automation*", in G.Kadoda (Ed). Proc PP1G 2001 pp85-103.

[Brumitt 00] 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

[Callaghan 04] 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

[Callaghan 07] Callaghan,V., Chin, J.S.Y., Shahi, A., Zamudio, V., Clarke,G.S., Gardner,M., *'Domestic Pervasive Information Systems: End-user programming of digital homes'*, Journal of Management Information Systems, Special Edition onPervasive Information Systems (volume editors Panos Kourouthanassis; George Giaglis), 24:01 (December 2007) pp.129-149 (inc), ME Sharp (New York), ISBN: 978-0-7656-1689-0

[Callaghan 08] Callaghan V, Clarke G, Chin J "*Some Socio-Technical Aspects Of Intelligent Buildings and Pervasive Computing Research*", Intelligent Buildings International Journal, Vol 1 No 1, Published Autumn 2008, ISSN: 1750-8975, E-ISSN: 1756-6932

[Chen 04] 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

[Chen 00] G. Chen and D. Kotz. "*A survey of context-aware mobile computing research*", Paper TR2000-381, Department of Computer Science, Darthmouth College, November 2000.

[Chin 08] Chin J, Callaghan V, Clarke G, "*A Programming-By-Example Approach To Customising Digital Homes*" IET International Conference on Intelligent Environments 2008, Seattle, 21-22 July 2008

[Chin 09] Chin J, Callaghan V, Clarke G, "Soft-appliances: A vision for user created networked appliances in digital homes", Journal of Ambient Intelligence and Smart Environments 1 (2009) 65–71, DOI 10.3233/AIS-2009-0010, IOS Press, 2009

[Chung 03] 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

[Cook 03] D.J. Cook, M. Huber, K. Gopalratnam and M. Youngblood, "*Learning to Control a Smart Home Environment"*, Innovative Applications of Artificial Intelligence, 2003

[Cook 06] Diane J. Cook, Michael Youngblood and Sajal K. Das "*A Multi-agent Approach to Controlling a Smart Environment*", pp 165-182, in Designing Smart Homes: The Role of Artificial Intelligence, ISBN 978-3-540-35994-4, July 2006

[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

[DiDuca 05] DiDuca D and Van Helvert J. "*User Experience of Intelligent Buildings; A User-Centered Research Framework*", Intelligent Environments 2005, Essex, 28-29th June 2005

[Dulay 05] N. Dulay, S. Heeps, E. Lupu, R. Mathur, O. Sharma, M. Sloman and and J. Sventek, "*AMUSE: Autonomic Management of Ubiquitous e-Health Systems*", Proc. UK e-Science All Hands Meeting, Nottingham, Sept. 2005

[Duman 07] Duman H, Callaghan V, Hagras H, "*Intelligent Association Selection of Embedded Agents in Intelligent Inhabited Environments*", Journal of Pervasive and Mobile Computing, vol. 3, issue 2, 2007, 117-157

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

[Halbert 93] Halbert D.C. "*SmallStar: Programming by Demonstration in the Desktop Metaphor*", Watch What I DO, MIT Press. 1993

[Haarslev 01] V.Haarslev and R. Moller, "*Description of the RACER system and its application*", In proceedings International Workshop on Description Logics (DL-2001), 2001

[Hague 03] Hague, R., et al: "*Towards Pervasive 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

[Johnson 08] Johnson B, Callaghan V, Gardner G "*Bespoke Appliances for the Digital Home*" IET International Conference on Intelligent Environments 2008, Seattle, 21-22 July 2008

[Kainulainen 06] Kainulainen L "*Reasoning in The Smart Home*", AIOME, 2006

[Kidd 99] C. Kidd, R. Gregory, A. Christopher, T. Starner. "*The Aware Home: A Living Laboratory for Ubiquitous Computing Research*", In *Proceedings of the Second International Workshop on Cooperative Buildings (CoBuild'99)*, October 1999

[Lieberman 99] Lieberman H., Bonnie A. Nardi and David J. Wright, "*Training Agents to Recognize Text by Example*", Proceedings of the third annual conference on Autonomous Agents, Seattle, Washington, United States, 1999, pp 116 - 122

[Lieberman 01] Lieberman H, "*Your wish is my command", Program by Example*", Morgan Kaufmann press, 2001.

[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

[Luck 03] M. Luck et al., "*Agent Technology: Enabling Next Generation Computing*", AgentLink II, 2003

[Mäyrä 06] 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

[McCann 04] J. McCann, P. Kristoffersson and E. Alonso, "*Building Ambient Intelligence into a Ubiquitous Computing Management System*", Proc. International Symposium of Santa Caterina on Challenges in the Internet and Interdisciplinary Research (SSCCII-2004), Amalfi, Italy, Jan. 2004

[McDaniel 01] McDaniel R., "*Demonstrating the hidden features that make an application work*", in "Your wish is my command: programming by example", Morgan Kaufmann Publishers Inc. San Francisco, CA, USA , 2001, pp 163 – 174

[Minar 99] N. Minar, M. Gray, P. Maes. "*Hive: Distributed Agents for Networking Things*", In Proceedings of ASA/MA'99, the First International Symposium on Agents Systems and Applications and Third International Symposium on Mobile Agents, 1999

[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, and constraints*", ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 12 , Issue 2 (April 1990), pp 143 – 177

[O'Neill 04] O'Neill, E. and Johnson, P. (2004) "*Participatory Task Modelling: users and developers modelling users' tasks and domains*", 3rd International Workshop on task models and diagrams for user interface design (TAMODIA 04). Prague, Czech Republic, 15-16th November 2004

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

[Zamudio 08] Zamudio V, Callaghan V, "*Facilitating the Ambient Intelligent Vision: A Theorem, Representation and Solution for Instability in Rule-Based Multi-Agent Systems, Special Issue Section*" International Transactions on Systems Science and Applications (special issue on "Agent based System Challenges for Ubiquitous and Pervasive Computing"), Vol 4, No. 1, 2008

[Röcker 04] 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

[Schilit 94] B. Schilit, N. Adams, and R. Want. "*Context-aware computing applications*", In Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, 1994

[Smith 77] Smith, D. C., "*Pygmalion: A Computer Program to Model and Stimulate Creative Thought*", (1975 Stanford PhD thesis), Birkhauser Verlag. 1977.

[Sugiura 98] Sugiura A,Koseki Y. "*Internet scrapbook: automating Web browsing tasks by demonstration*" Proceedings of the 11th annual ACM symposium on User

interface software and technology, San Francisco, California, United States, 1998, pp.9-18

[Truong 04] 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.

[Vastenburg 04] Vastenburg M, "*SitMod: a tool for modelling and communicating situations*", Second International Conference, PERVASIVE 2004, Vienna Austria, April 21-23, 2004, ISBN: 3-540-21835-1

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

[Zamudio 08] 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

[Zou 04] Young Zou, Harry Chen, and Tim Finin, "*F-OWL: an Inference Engine for Semantic Web*", Proceedings of the Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems, 26 April 2004

Web Listings:

[AgentSheets] http://agentsheets.com/products/index.html
[HP Jena] http://jena.sourceforge.net/
[Network Inference] http://www.networkinference.com/Products/Construct.html
[PBDCAD] http://www.lisi.ensma.fr/ihm/index-en.html
[SOUPA] http://pervasive.semanticweb.org/soupa-2004-06.html
[Stagecast] http://www.stagecast.com/creator.html
[ToonTalk] http://www.toontalk.com/
[DCOMP] http://iieg.essex.ac.uk/dcomp/ont/dev/2004/05/
[OWL-S] http://www.w3.org/Submission/2004/07/