In Journal of Management Information Systems, Special Edition on Pervasive Information Systems (volume editors Panos Kourouthanassis; George Giaglis), 24:01 (December 2007) pp.129-149 (inc), ME Sharp (New York), ISBN: 978-0-7656-1689-0ß

JMIS *Journal of Management Information Systems*

Advances in Management Information Systems (AMIS)

Published: *ME Sharpe, New York, 2007*

Pervasive Information Systems

Edited by: *Panos E. Kourouthanassis and George M. Giagli*

CHAPTER 5
Domestic Pervasive Information Systems: End-user programming of digital homes

Vic Callaghan, Jeannette Chin, Victor Zamudio,
Graham Clarke, Anuroop Shahi and Michael Gardner
Email: lifestyles@essex.ac.uk URL: http://digital-lifestyle.essex.ac.uk

Abstract
The chapter presents the background to the development of the digital home of the future and the ways in which it might be controlled by the end-user. We describe the technical background to the development of the digital home out of the ubiquitous availability of networks and devices. We then describe two different approaches to user control that are already under development – Task Based Computing (TBC) and Pervasive Interactive Programming (PiP). We discuss theoretical work on combining, formalising and visualising these processes. In addition we report on a user evaluation that demonstrates that non-expert users find these methods to be simple, enjoyable and useful. Although this chapter confines itself to end-user programming of the digital home we argue that the underlying mechanisms and concerns apply to all levels of pervasive computing.

Keywords
end-user programming, task computing, digital home, smart home, intelligent buildings

1. Introduction

In "Towards A New Architecture", a ground breaking text of the Modern Movement in Architecture in the 1920's, Le Corbusier famously remarked that, "*A house is a machine for living in*" (Corbusier). More recently Craig Mundie, one of three chief technology officers at Microsoft, the world's largest software company was quoted in the Economist as saying "We view the digital home as critically important" and "the home is much more exciting than the workplace." Microsoft are not alone as, for example, Intel, the world's largest semiconductor maker, was reported in the same article as reorganising itself into new business divisions including one called the "digital home" (Economist 2005). The importance of the home market is reinforced further by market research by the Diffusion Group (TDG) which reported that in 2005, more than half of US households were interested in some sort of home control system (DTI 2005). Modern buildings have strong physical similarities to machines in that they contain a myriad of sensors, effectors, computer based devices and networks. By facilitating programmed coordination and interaction between distributed computer-enabled networked appliances, sensors and actuators the so-called smart home is created in which the home senses the actions of people and responds in programmable ways. Thus, an essential aspect of a smart or digital home is "programming". We will describe the technical background to the development of smart homes and then look at different ways in which such systems might be programmed. We will argue that personal choice and control is of the essence when it comes to choosing a programming approach. We will describe a number of scenarios of a possible digital home of the future to illustrate the approach we have adopted. We will then look in detail at two existing approaches – Task Based Computing (TBC) and Pervasive Interactive Programming (PiP) – and discuss the implications of these approaches for the underlying methods required to enable this. PiP has been evaluated by a number of users and the results of these finding will be reported.

2. Background to the development of the digital home

In terms of domestic homes, the roots of building automation can be traced to a small Scottish company, PICO, who, in 1975, started the X10 project which in 1978 resulted in Radio Shack introducing X10 home-automation technology to the American market. The X10 standard enables a computer, with suitable software, to control electrical power outlets by propagating signals along the power line. However X10 has its limitations, such as speed (it takes about 600ms to send a single command), collisions (simultaneous signalling causes the system to fail), signal strength (poor or noisy wiring environments cause failure) and limited addressing range (256 addressable modules, based on 16 house codes (A - P) and 16 unit codes). As a consequence there are numerous newer standards (e.g. LonTalk, BatiBus, CEbus, EIB, EHS, HBS etc) which seek to overcome these constraints and to expand the applications beyond simple actuator and sensor I/O into areas such as media streaming and interaction with internal functions of appliances (Wacks 1998). The arrival of the Internet in the early nineties and broadband networking for the home at the turn of the millennium has also impacted upon the functions and performance of home networks. This improved functionality has meant that the home computer is no longer just a gateway for the home, but can now be a contender for the home control network. Only time will tell which of the many standards will eventually dominate the domestic market but, for the time being, the simple and low-cost nature of X10 means it remains one of the most enduring standards (Adair 2005). Home automation standards are essentially descriptions of network transport mechanisms and communication protocols.

3. Programming the Home of the Future

The main focus of this chapter is on how the digital home of the future can be programmed and managed by ordinary non-expert home occupants. An alternative approach involves the use of autonomous intelligent agents that monitor an occupant's habitual behaviour, learning their needs, creating rules (self-programming) so they can pre-emptively set the environment to what they anticipate the user would like (Callaghan et-al 2005). Whilst autonomous agents may appeal to many people, their acceptance is not universal. Some lay-people distrust autonomous agents and prefer to exercise direct control over what is being learnt, when it is being learnt and to whom (or what) any information is communicated. These concerns are particularly acute when such technology is in the private space of our homes. Often, end-users are given very little, if any, choice in setting-up such systems, but rather, they are required to "surrender their rights" and "put-up-with" whatever is provided (Chin et-al 2004). Moreover, there are other good reasons in support of a more personal involvement, such as enabling the creative abilities of people by providing them with the means to become 'designers' of their own "pervasive computing spaces", whilst at the same time, shielding them from unnecessary technical details. In this approach, if people are given the means to configure their own 'electronic spaces' then personal expression will be able to extend beyond the current DIY approach of "paint and wallpaper" into information and control spaces. To achieve this vision it is necessary to solve the formidable challenge of enabling non-technical people to program coordinating sets of pervasive computing based home appliances. Current end-user programming systems for the home are built around extensions of the principles of conventional computer languages involving a sequence of logical instructions. In an attempt of making the process simpler for non-technical users, some applications, such as ActiveHome Pro (Asaravala 2004), employ a graphical interface front-end approach which represents text-based program constructs (i.e. instructions) with graphical objects for the user to manipulate into program flows or sequences of actions. The disadvantages of this approach are that, it requires users to mentally manipulate programming abstractions, is restricted to sequential actions (macros) and it is limited to single monolithic processor control rather than the distributed computation afforded by pervasive computing. The remainder of this chapter will present new research aimed at enabling non-technical home users to control and program distributed pervasive home networked devices in as unconstrained a way as possible.

2.1 An Illustrative Scenario

The following scenario is offered to crystallise some of the ideas discussed in this chapter. We will refer back to this scenario later in the paper when discussing different techniques.

1. **Background** - *Tessa is a visiting researcher at the University of Essex. She arrives at the University and moves into her new temporarily accommodation, an intelligent apartment. Like all environments in the future the 'radio-sphere' is awash with services that are available for her use. Many of these services are local such as lighting and, heating whilst others are remote such as video, music, news, email. Monolithic appliances and computer applications have given way to more atomic networked functions such as switches, video displays, codecs, editors, mp3 files etc.. Tessa interacts with the environment via her personal 'wireless assistant' (WA) which also holds descriptions of her preferred world.*
2. **Virtual Appliances & Applications** – *The concept of appliances and applications has lingered on as people still need to utilise functions akin to TVs, telephones, word processors etc. Consequently all environments have their networked devices / applications pre-formed into familiar default configurations (called Meta-Apps (MAps).) Each MAp describes a familiar everyday appliance. Thus, both physical and information spaces function as normal. It is possible for users to purchase new MAps and, for more creative individual, to devise their own.*
3. **Mobility** - *On entering her apartment, Tessa's WA starts to flash in an unobtrusive manner indicating she is within a 'smart space'. Her WA contains her ontology-based descriptions of her preferred MAp, It discovers what is available in the environment, and then requests that as near matches as possible be constructed. If devices move out of the room or fail, the system will similarly try to find suitable replacements. Of course this is not always possible but her WA will indicate what is missing, so she has the option to borrow, buy or replace any missing devices. One such MAp is her 'communication centre' (CC). On moving to other rooms and environments the WA attempts to maintain Tessa's preferred configuration for her CC MAp.*
4. **Programming** – *The original CC MAp consists of a telephone service, audio transducer and dialler. Tessa has modified the MAp to add in a light and then programs rules using an end-user programming tool that is resident in her WA to be associated with this new device. For example she re-programmes the CC MAp configuration and rules to, "on receipt of a call, pause other incoming media streams, divert the call to the audio/video-transducer in use at the time, and raise the light if it is dark". While Tessa generally only modifies existing MAps there*

are numerous hobby clubs and small industries that generate novel and sometimes highly complex MAps which they then trade.

5. **Interaction -** *Tessa selects the 'News' menu, which causes the smart space to invoke an interactive display MAp, connecting it to her preferred RSS News feeds. Whilst reading her news feed, a video-conference request arrived, and the CC acts like a sophisticated 'soft-appliance', activating previously programmed rules that cause the news feed to be suspended, lights to be raised and the video conference to be patched through to the current audio and video system. As with a normal appliance, Tessa can manually override any of the settings on this "soft-appliance".*

From this scenario it can be inferred that, in order to realise this particular vision a number of issues need to be resolved. These include the question of how communities of devices are formed and managed, how capabilities of devices and communities are described, how lay users can program these communities, how the system deals with mobility of devices and users, how the user interacts with the programmed systems and how the end-user can maintain and debug the system. We will hope to answer these questions in the process of describing this approach further.

2.2 De-composition, De-construction, Dis-integration and Dis-aggregation

Whilst traditional stand-alone home appliances provide useful functionality to users, when you add a network connection a number of significant possibilities arise. For instance, manufacturers could provide access to individual sub-functions within an appliance or application allowing, for example, the mute function on a TV to be accessed by other networked appliances. More significantly *soft-appliances and applications* could be created by establishing logical connections between sub-functions. This might be used to create replicas of traditional appliances and applications, or to invent altogether new appliances or applications (Chin 2005). In essence this paradigm involves the *deconstruction* (alternatively described as decomposition, dis-aggregation or disintegration) of traditional appliances and applications into their atomic functionalities (physically or logically), allowing the user to re-construct appliances and applications by reconnecting the basic atomic functionalities in various ways. Some current examples of this approach include SUN's Epsilon Project (Epsilon 2005), which is exploring how appliances are decomposed into small independent devices each having a virtual world proxy which can be "connected" to other proxies to create meta systems (offering conventional appliance functions, or novel ones created by user chosen combinations). A particularly interesting aspect of the Epsilon work is that it explores the notion of ultra-thin clients where the physical manifestation of the appliance becomes near stateless with most state and process residing on proxies whose location is almost irrelevant. This work at SUN is wide ranging and includes studies on supporting middleware (Horan 2005). As part of their EasyLiving project Microsoft are also exploring the notion of deconstruction (dis-aggregation in their terminology) to PCs and services, demonstrating how a disconnected pool of screens, keyboards and applications can be dynamically re-connected to create a virtual PC for a user in differing contexts (Easy Living 2005). In terms of decomposed applications, Apple's Unix Mac OS X (aka "Tiger") provides an 'AppleEvent subsystem' that allows developers to get at the internal interface descriptions of applications (i.e. application sub-functions) and combine them in differing ways (Jobs 2004).

2.3 Communities and Tasks

The key to creating soft-appliances and applications from deconstructed functions is connecting them into coordinating communities or collectives, synergistically forming new functions. Clearly, the richer the pool of (sub-) functions or services, then the greater the possible permutations for new utilities. How such communities or collectives are created is one of the central issues that we address in this chapter.

A useful way to view people and their activities, which is consistent with the deconstructed world view we are developing, is to see their activity as being task based. For example, rather than describing user requirements in terms of the physical model of the world - "I will switch on the TV in the corner of the living room, and turn to channel 3" - one might abstract to the higher level task "I want to watch the news now (where I am)". Later in this chapter we will show how such a task based approach can be implemented to provide a user-friendly means of interacting with home based pervasive information systems.

2.4 Making Sense of the World: Ontology and Epistemology

In order for the tools we provide the user to make sense of their world, it is necessary to develop a description of the properties and capabilities of devices and applications that can be shared. An ontology formally describes devices and applications, and provides axioms that constrain the form and interpretation of these terms. An ontology could therefore help with mobility and failure by searching for nearest matches to missing devices or community functions, or by alerting the user to other possibilities given the particular context. For user generated communities of coordinating computer based devices, the community related information can be described and reasoned about using an ontology. Ontologies also provide a convenient means for storing rules which embody the autonomous functionality of a community. Since we are concerned with user defined communities that are both personal and subjective descriptions based upon the limited knowledge of the user their representation within an ontology is referred to here as an epistemology. As the techniques described in this chapter are user-centric, epistemologies are intrinsic to many of the approaches we describe and are discussed in greater detail in a later section.

3. Task Based Computing (TBC)

Task based computing was pioneered by Wang and Garlan of CMU (Wang & Garlan 2000) and Fujitsu (Masuoka et-al 2003). It seeks to provide a programming environment that allows users to interact with computing spaces in terms of high level tasks. It can be viewed as a method to allow users to discover, combine and execute coordinated contextual actions (tasks) which differs from the more common usage such as for capturing system requirements and for specifying users interfaces (O'Neill & Johnson 2004). Thus, in our interpretation, tasks are high-level collectives composed of numerous lower-level actions, for example the task "Play My MP3 Files" could be decomposed into a series of smaller steps which need to be combined to carry out this task. In the Fujitsu work, a GUI tool referred to as STEER (Semantic Task Execution Editor) is used to do this. The basic unit of task composition is a pair of service inputs and outputs which, when associated, can be executed on command by the user. Using STEER is possible to create more complex compositions. This approach can be seen to have similarities to scripting mechanisms (e.g. AppleScript) that enable the user to combine the functionality of multiple applications. For example, Apple's "drag & drop" Automator tool allows developers to create lists of actions (workflows) in new and even unexpected ways (Jobs 2004). In general terms, TBC provides a simple and quick way for users to interact with and control such environments since they simply need to select required actions from a menu of available high-level tasks with a minimal of configuration and interaction.

3.1 Task Discovery

Pervasive computing environments, such as smart spaces, contain a range of services which are resources provided to network clients by one or more servers like a room light for instance. In this implementation, a service keeps no record of its own or its client's state and does not have to provide a unique identifier. However, tasks are normally constructed from a set of services and keep state on themselves and clients. Before interacting with services, some form of service discovery is necessary, which should be seamless and intuitive.

```
<taskgroup label="IIE Room">
    <taskgroup label = "Control Space">
        <taskgroup label = "Lighting">
          <task label="Switch On" target="http://essex.ac.uk/idorm#LightOn"
           oncomplete="Let there be light"/> </taskgroup>
    </taskgroup>

    <taskgroup label = "Personal Space">
        <task label="News" target="http://essex.ac.uk/idorm#NEWS"/> </taskgroup>

    <taskgroup label = "NoticeBoard">
       <task label="Add Note" target="http://essex.ac.uk/idorm#ADDNOTE"
        oncomplete="Enter NOTE on Board"/> </taskgroup>
</taskgroup>
```

Figure 1 - Task Definition For Grouping Tasks Available In The IIE Room

With a user orientated task layer, pervasive computing environments are able to discover and present combinations of services to users as high level tasks, which may be organised according to contextual information. Figure 1 shows a task definition describing tasks available within the iDorm. Tasks are grouped according to context-based namespaces, such as ‘IIE Room/Control Space/Lighting/Switch On’. This namespace indicates that ‘Switch On’ is an atomic task for controlling lighting in the IIE Room. For this simple case, the low level service (switch on) directly equates to a task (i.e. one source, one sink). In such simple cases, services will belong to well defined types, allowing task definitions to be generated by pairing sources and sinks, inferring the type of task by making use of a service’s low-level interface description, as provided by conventional service discovery protocols. Where multiple services are combined to form tasks, they can be organized automatically using available contextual information, without requiring any human intervention or being organised manually by the user. In this work, tasks are provided to the user via a smart phone device using Bluetooth; Figure 2 depicts the results of transferring tasks described by Figure 1 to such a phone which a user may then use to discover and interact with the pervasive information systems environment.

Figure 2 – Task Menu on Smart Phone

The automated approach is crucially dependent on designers being able to pre-specify all combinations of device types and users, and developers adhering rigidly to type constraints. Thus, for more complex collectives involving numerous sources, sinks, conditional relationships, and/or user created communities, automated task formation is infeasible. In such cases a method for translating composite sets of services into tasks is required. Pervasive Interactive Programming (PiP) is one such tool and is described later in this chapter.

3.2 Task Interaction

Task based computing shields users from knowing esoteric details associated with a service's interface definition. This is provided by linking a task definition to a service's interface definition (Figure 1). Figure 3 depicts an architecture describing a task based computing environment where a mobile device, such as a smart phone, is present within a space, and is interacting with a lighting service provided by the space.

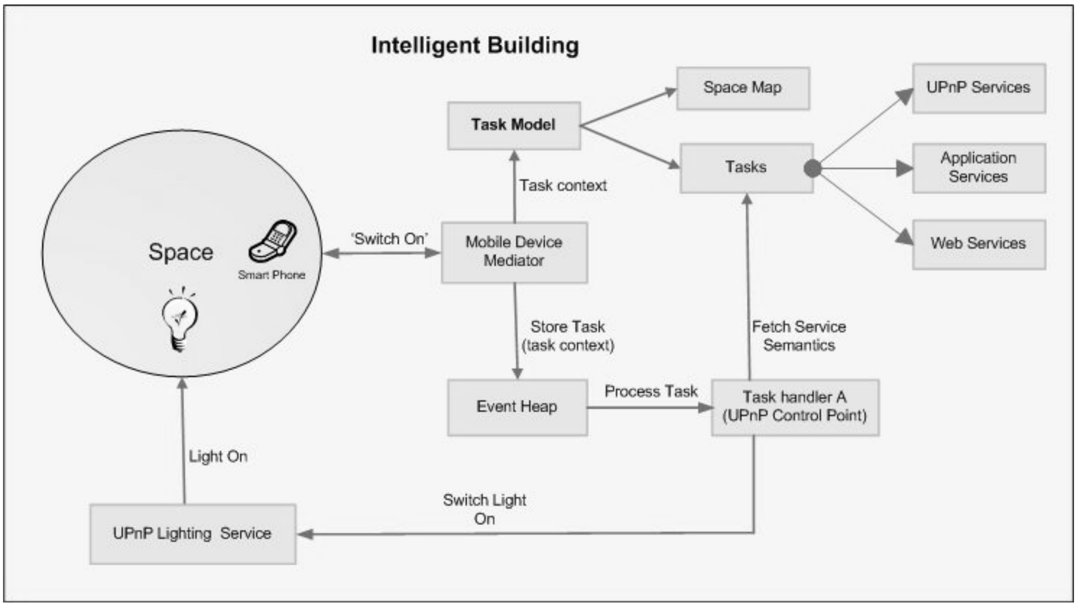

Figure 3 - Task Interaction in a Pervasive Computing Environment

The architecture is divided into a number of components:

- **Mobile Device Mediator** (MDM): Provides mechanisms for discovering and interacting with any services available. Interacts with the task model to obtain a task definition (figure 1), which is then translated and adapted to a form interpretable by the device. When a user invokes a task, such as 'Lighting On', the MDM forwards the request to the event heap.

- **Task Model** (TM): Stores all task definitions and their mapping to task descriptions. Organizes tasks according to user-defined notions of space. For example, one may wish to split a room into lots of sub- spaces or tag objects using RF-ID, consider a whole floor using Wi-Fi, or just a room by using Bluetooth. This will associate tasks with physical spaces within a building.

- **Tasks**: Abstracts environmental and application services into processes as defined by the OWL-S. Tasks are embedded with semantics to allow for automated and seamless service composition, invocation and configuration. Tasks may correspond to either atomic processes or composite processes. For example, in Figure 1, the task 'Lighting'/'Switch On' is mapped to an atomic service identified by http://essex.ac.uk/idorm#LightOn which, in turn, links to a semantic description that wraps an operation from a UPnP based lighting service.

- **Event Heap and Handlers**: All task invocations received from the MDM are forwarded to the event heap, which then notifies any relevant task handlers registered for a particular task type. The 'task invocator' handler processes events by using the task to invoke a relevant service, e.g. a UPnP based lighting service.

Prior to their use, simple tasks are generated from automated parsing, but more complex collectives need to be explicitly programmed.

4. Pervasive Interactive Programming (PiP)

In this section we describe an end-user tool that takes the notion of task based computing forward by enabling the creation of non-terminating tasks or meta-appliances (and meta-applications) from locally available appliances (and application) services. These non-terminating tasks can be programmed by the user.

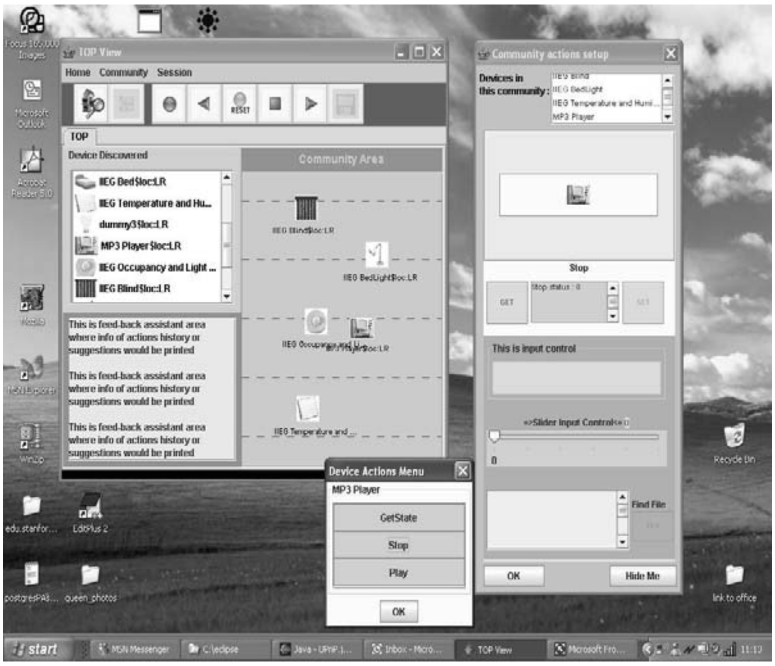

Figure 5 – The MaP Editor With Example Community Set-Up

4.1 Overview

Pervasive Interactive Programming (UK Patent No: GB 0523246.7) is based on the idea of putting the end-user at the centre of control of a pervasive information system environment by providing a simple means, requiring no technical skills. This approach allows the user to define communities of pervasive devices and to program them by using this community to produce the required behaviour. Such coordination creates behaviours above and beyond those available from an individual application or appliance giving rise to a possible alternative name for the process Meta Application-Appliance Programme (MAp). PiP has its roots in Programming-By-Example (PBE), an programming paradigm pioneered by Smith in the mid-seventies where functionalities are not described abstractly but rather demonstrated in concrete examples (Smith 2000, Canfield et-al 2000, Lieberman 2001); Tangible Computing, a way of bringing a physical metaphor to software abstractions pioneered by Ishii (Ishii et-al 2004); Palpable Computing (Andersen et-al 2005), an approach to promote user control and choice through increased visibility of pervasive computing technology, and Learning-From-the User (LFU), an embedded-agent learning paradigm Essex University has been developing for many years (Callaghan 2005). In addition, PiP utilises ontologies mainly drawn from research work on the semantic web (Berners-Lee et-al 2001). PiP differs from PBE in that, firstly it aims at real rather than graphical objects, secondly it is directed at distributed computing rather than a single processor, and thirdly it spawns distributed non-terminating sequence independent tasks rather than creating macros or other procedural structures. PiP shares the same motivation as many of the above mentioned approaches but aims to take this vision forward by enabling non-technical people to become designers and programmers of their own unique environments. In addition, the motivation for PiP was driven by experience with autonomous agent based systems where concerns about privacy and trust were voiced (Callaghan et-al 2006, Basu & Callaghan 2005, Chin et-al 2004, Lyons 05). In the PiP approach, the system is explicitly put into a learning mode and is taught how to behave by the end-user demonstrating the activity required. For example, as discussed in the scenario provided earlier, the TV or sitting room light could be made to react to an incoming call on the telephone, thus the telephone, TV and light could coordinate their actions to form a new meta-utility (soft-appliance). In this approach functional sub-units of appliances can be shared while devices interoperate seamlessly together. For example, the audio amplifier in a TV could be made use of by the HiFi system, or vice versa. Consequently, MAps could be created by establishing logical connections between the sub-functions of appliances, creating replicas of traditional appliances, or inventing altogether new appliances. Of course there are also stand-alone appliances that provide all these functions in an "off-the-shelf" box. Additionally, the vision for PiP includes the notion of pre-fabricated interconnection MAps which are descriptions of previously made communities, such as a TV.

4.2 PiP Architecture

PiP supports setting up communities. The user first selects and defines a community that they wish to program, and then carries out a set of coordinated actions that are taught to the system. Those members of the community that are going to be the actuators and those that are going to be the environment for such actuation need to be chosen. A user action in the teaching mode causes an appliance to generate an associated event, and this event is then used to generate appropriate rules based on a "snapshot" of the environment (community) state at the time. A device can be involved in more than one community. The user interface with PiP is via a PiP editor, shown in Figure 5. This editor provides a means for:

(1) displaying discovered devices,
(2) setting up / amending communities and
(3) managing user's demonstration sessions (teaching).

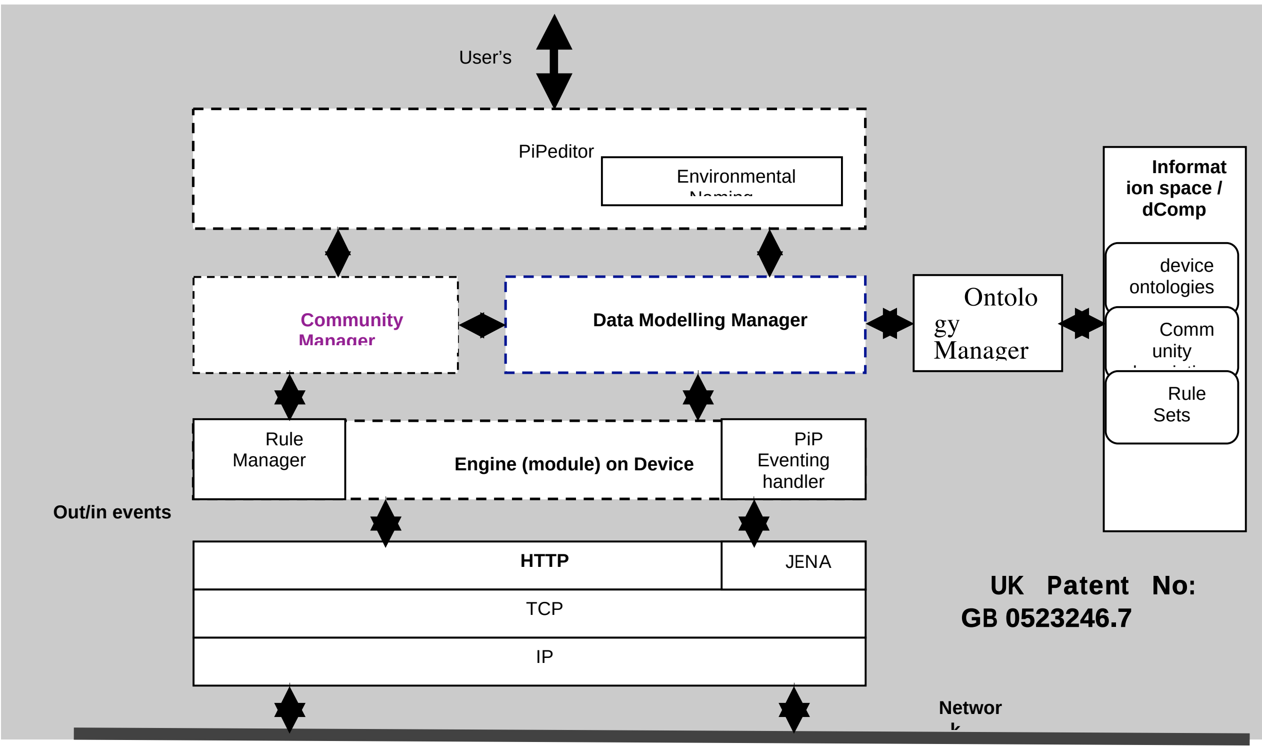

Figure 4 - The PiP Architecture

Tasks (e.g. MAps) can be taught from either interacting with on-screen representations of the devices or with the real devices themselves. Once created, tasks can be played back on-demand, either from the user generated event, as in a Task Computing meta-application, or in response to a environmentally originated event.

3. Community Manager— this module manages (sets up and maintains) the communities of coordinating devices.
4. PiPeditor – this is the interface the user uses to program and interact with the system.
5. RuleManager – this module responsible of compiling and executing rules.
6. OntologyManager – this module manages the translation of ontology

To facilitate the information to be used within and beyond the community, data needs to be standardised so that it can be understood by all other parties in the network. For this aspect of work, the semantics in PiP (described in the following section) supports information interoperability between applications, providing a common machine "understanding" knowledge framework.

4.3 Semantics of Home Devices and dComp Ontology

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

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

Table 1 - dComp ontology (v.1.1)

The SOUPA ontology from Ubicomp (Chen et-al 2004) is aimed at pervasive computing but lacks support for crucial PiP mechanisms such as community, decomposed functions and coordinating actions which are essential to produce higher level meta functionality. In addition, the current SOUPA standard has only limited support for the UPnP standard (which our research testbed, the iDorm2, which is described later in this chapter depends on). OWL-S, previously called DAML-S (OWL 2005) is based around the notion of services. It primarily targets the World Wide Web, enabling agents to evoke services thereby facilitating the automation of web tasks. It provides a useful abstraction called composite processes which is still under development and, at the time of writing, does not give a precise specification of what it means to perform a process. Thus, we have developed our own ontology, dComp (deconstructionist & community programming) that provides a better match to domestic environments and has a well defined specification of communities (akin to composite processes on OWL-S). dComp (see Table 1) is based around the OWL language which is widely used (especially for the semantic Web). There are also numerous supporting

In Journal of Management Information Systems, Special Edition on Pervasive Information Systems (volume editors Panos Kourouthanassis; George Giaglis), 24:01 (December 2007) pp.129-149 (inc), ME Sharp (New York), ISBN: 978-0-7656-1689-0ß

tools such as the Jena (McBride 2001), RACER (Haarslev & Moller 2001) and F-OWL (Zou et-al 2004) inference engines are widely available. The full dComp specification is available online (dComp 2005).

In the current implementation we have defined a few classes in supporting the notion of community and rules (Chin 2005). Wherever possible we have sought to adopt suitable ontology for our other needs. For example our Person, Policy and Time ontology are adopted from Ubicomp SOUPA ontology. In dComp, preferences are referred as ‘situated preferences’, which is akin to Vastenburg’s ‘situated profile’ concept where he uses situation as a framework for user profile so that the values of the profile are relative to situations (Vastenburg 2004). By way of an illustration of a virtual appliance definition, figure 6 shows a partial description of a TV community. In this, the community has a label “JC TV” with a description of “The first JC testing TV”, it was created on the 2004-09-06 at 19:43 and has an owner “Jeannette”; it was composed from 3 other devices on the network. These devices are identified by their unique id numbers were: UUID:PHLCDT17, UUID:PHLAudioMMS223, and UUID:NetGem442.

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

Figure 6 – a partial TV Community Definition

5. Agent Services

At the outset of this chapter we described how automated services, such as agents were deliberately made to be subservient to the end-user programming interface, allowing the user to choose the level of autonomy given to various parts of the system. Invariably there are some aspects of any system that need to be automated as users will not be competent to carry out or want to be involved at the level of complexity or abstraction involved. For example, searching the available network services and functions, and mapping these to higher level task based user requirements, would be a complex and tedious process that is better left to automated assistance. This difficulty is compounded by the highly dynamic nature of users and devices, with both users and devices joining and leaving networks, and the variability of devices, it is not possible to pre-specify every device or combination of devices. In the following we provide a formalism that describes how continuity of function might be supported when people and devices change in ways illustrated in the Home of the Future scenario earlier in this chapter.

In general terms we could describe this problem as follows: an ***allocation*** is a duple (d,T) where d is a device and T is a not empty set of k tasks, *i.e.* $T = \{t_1, t_2, t_3, \ldots, t_k\}$, with $k \geq 1$. If $k = 1$ we have a simple device, that is able to handle only one kind of task. This is the case of a speaker, or a microphone. If $k > 1$ then d is a complex device, which is composed by other sub-devices, ***i.e.*** d can handle more than one task. This could be the case of a TV, composed by a device that can handle two different kinds of signals: audio and video.

When the user configures a new appliance, he defines a new ***community***. A ***community***, denoted by C, is a finite not empty collection of n allocations, *i.e.*

$$C = \{(d_1, T_1), (d_2, T_2), (d_3, T_3), \ldots, (d_n, T_n)\}, \text{ with } n \geq 1 .$$

If the user goes to a new environment, the agent should attempt to create an ***equivalent community*** C_{eq}. In order to create this equivalent community, for each allocation $(d, T) \in C$ the agent should find an equivalent allocation (d_{eq}, T_{eq}) in the new environment. As we mentioned before, we have two cases: $k = 1$ and $k > 1$.

i) If $k = 1$ then d is a simple device and $T = \{t_1\}$. The agent should find a new allocation $(d_{eq}, \{t_1\})$ such as the device d_{eq} is able to perform the only task t_1.

ii) If $k > 1$ then d is a complex device, and $T = \{t_1, t_2, t_3, \ldots, t_k\}$. The agent should find, in the worst case, k allocations $(d^1_{eq}, \{t_1\}), (d^2_{eq}, \{t_2\}), (d^3_{eq}, \{t_3\}), \ldots, (d^k_{eq}, \{t_k\})$, where every device d^i_{eq} is able to perform the task t_i, with $1 \leq i \leq k$.

We could extend this framework in order to include time. A ***temporal allocation*** is a tuple (d, T, t_i, t_f) where d is a simple device, T is a (simple) task, t_i is the initial time and t_f is the final time. In other words, the device d will be performing the task T during $t_f - t_i$ units of time, beginning on the instant t_i. So, a ***temporal community,*** denoted by C_t is a non-empty set of temporal allocations:

$$C_t = \bigcup_{j=1}^{k} \{(d_j, T_j, t_{ji}, t_{jf})\}$$

From this approach other issues arise, such as scheduling and time-dependant sequences of tasks (Zamudio et-al 2005). A temporal community representation in agent service is shown in Figure 7.

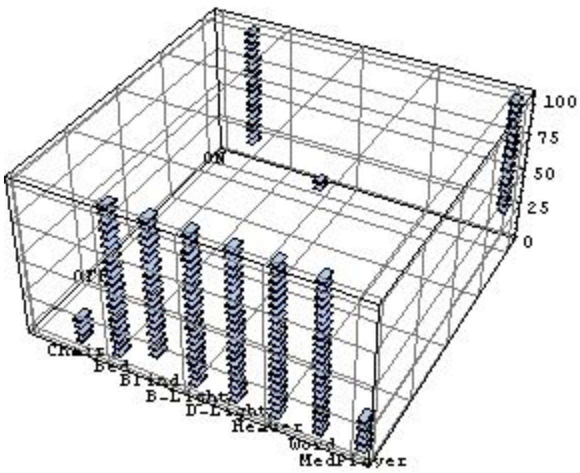

Figure 7. Representation of a temporal community

6. Evaluation

6.1 Research Platform

For our experimental work we have built a pervasive computing test-bed called the iDorm2 which takes the form of a two bed roomed apartment see Figure 8. It is a full size domestic apartment containing the usual rooms for activities such as sleep, work, eating, washing and entertaining. It comprises numerous networked artefacts such as telephones, MP3 players, lights, beds and chairs.

Figure 8 – The iDorm 2

Connectivity and a common interface to the iDorm2 devices are implemented via IP networking and Universal Plug & Play (UPnP). UPnP is a distributed middleware that employs event-based communication, supporting automatic discovery and configuration. Our experimental PiP architecture aims to be independent of any particular middleware although the current version utilises UPnP as its underlying network communication infrastructure. The PiP user interface can be accessed via a variety of means ranging from mobile devices such as tablet PCs to an LG iFridge.

6.2 Procedures and Apparatus

The work described in this chapter was evaluated based on a trial involving eighteen participants drawn from a diverse set of backgrounds (e.g. housewives, students, secretaries, teachers etc). Their genders were 10 females and 8 males with ages ranging from 22 to 65. The participants also formed a multicultural group including Asians, Europeans, Latin-Americans and Australians. All participants have some computing experience (i.e. they knew how to use a mouse). Whilst 20% of the participants had a very good knowledge of programming, 60% of them had none at all. For the evaluation sessions they were given five sets of devices (drawn from a set of lights, a telephone, smart sofa and an MP3 player). During the evaluation, no specific tasks were set for the participants but they were encouraged to use their imagination to create their own desired environment based on the devices available. The evaluation was preceded by a 20 minute training session.

The evaluation methodology was developed with the assistance of Chimera (a socio-technical research unit based on the BT Research Park at Martlesham Heath in Suffolk, UK) to assess the participants' subjective views on the usability of the system (DiDuca and Van Helvert 2005). It consisted of both observations and a questionnaire measuring attitudes over six usability dimensions shown in Table 2 (a higher rating score on the dimensions shows greater usability). Each of these dimensions consisted of a series of statements (from 2 to 4) and each statement offering a range of ratings (from 1 to 5). A higher rating score on the dimensions contributes towards the greater usability of PIP.

Figure 9: Some participants evaluating MaP

6.3 Results

In this limited space is not possible to present all the evaluation data or results therefore only highlights will be given to convey the general findings. What is clear that all the dimensions rated well (scoring above 4) indicating they users were generally well satisfied with the system. At the outset of the work, one of our contentions was that people would enjoy the experience of programming and find it relatively easy. Both of these assertions were supported by the evaluation as, in terms of enjoying the experience, the mean of the affective dimension was 4.6 (the highest rating) indicating people greatly enjoyed the experience of PiP programming, whilst the cognitive load dimensions had 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.

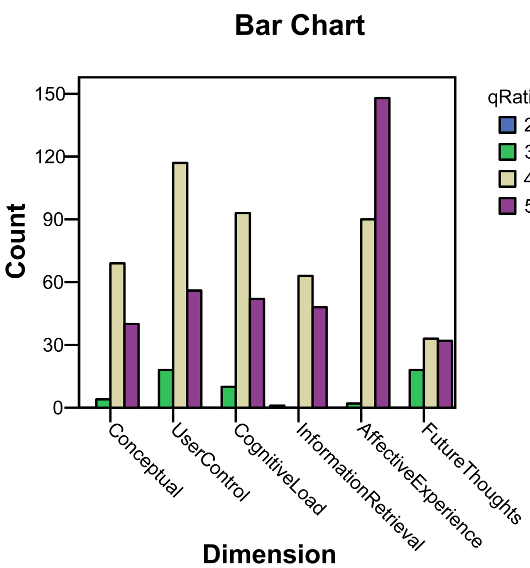

Figure 10 - The Six Dimension Rating

Although not shown in the data presented here, we found no significant variation across culture but found some minor variation on cognitive loading for age groups, with younger participants finding it the system slightly easier to use. In general the "Information Presentation" dimension (how well information was presented to the user) scored the least but was still in excess of 4 indicating that overall people found it usable. Given that this is an early prototype, we were not surprised to find that the interface could be improved. None of the participants found the principles difficult to understand. A remark from one participant stated "*I thought the basic principles themselves are very simple and straight forward. I felt I could easily grasp the basic principles*" was typical of many users. This particular comment was from the group with no programming skills at all (a key target of our work). Overall 83.4% of all participants found the system intuitive to use and 94.4% of all participants stated they felt it rewarding to use the system. Thus these initial results support the original thesis of the work that it is possible to produce a system that empowers non-specialists to be able, and to enjoy, programming coordinated actions of distributed embedded computer systems that form a crucial aspect of the digital home.

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean	
					Lower Bound	Upper Bound
Conceptual	113	4.3186	.53894	.05070	4.2181	4.4190
User Control	191	4.1990	.59134	.04279	4.1146	4.2834
Cognitive Load	155	4.2710	.57332	.04605	4.1800	4.3619
Information Presentation	112	4.4107	.54613	.05160	4.3085	4.5130
Affective Experience	240	4.6083	.50596	.03266	4.5440	4.6727
Future Potential	83	4.1687	.76221	.08366	4.0022	4.3351
Total	894	4.3602	.59489	.01990	4.3211	4.3992

Table 2 - The Evaluation Ratings

7. Discussion

7.1 Summary

In this chapter we have described how domestic pervasive systems of the future might be composed of potentially hundreds of coordinating deconstructed services and functions. Most of these will function in the same way as current appliances and applications, although their physical appearance might be very different to current products. We have described two complementary approaches to supporting non-technical user of future digital homes; task based computing (TBC) and Pervasive Interactive Programming (PiP). Both approaches are based on the notion of constructing atomic computational elements into higher level tasks. In PiP tasks are wrapped within the "appliance" metaphor, which is a well established idea in home environments, and are created by the user using real devices (or graphical emulations of them) to demonstrate what is required. The complexity and variety of tasks that can be programmed are limited only by the user's actions which provide both its distinctive edge and principal research challenge. In TBC tasks are created by the system which associates service providers and consumers that the system has found within the local environment, on the basis of pre-programmed services. PiP is able to extend the capabilities of TBC by providing a mechanism for users to create tasks that go beyond the limits of anticipated or pre-programmed use.

We have described an ontology (dComp) that allows meta appliances and applications to be defined and configured. These descriptions can be either supplied with systems of possible behaviours so that the devices offer a default functionality akin to current appliances and applications or, for the more creative end-user, a system that will enable them to create their own novel meta-appliances and applications (MAps), thereby allowing them to decorate their domestic environments in new ways, something we have dubbed '*DIY in the pervasive computing age*'.

We have provided a formalism that describes the task translation and allocation problem needed to support MAps and movement of people and devices. Finally, although this work is aimed at the future digital home rather than those existing now, we have built and evaluated a prototype 'proof of concept' system. Whilst we have only been able to undertake a comparatively small scale evaluation with 18 users the initial finding are most encouraging as they support our original hypothesis that it is 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.

7.2 Future Directions

Our longer term work will involve the refinement of techniques elaborated in this chapter. While we have directed our work at a domestic setting, we believe that these methodologies are generic in nature and can be applied to other environments, something we will pursue in the future. Another area we are especially interested in exploring is the synergy in the interoperation of an *ontology engine* in support of user based programming. How might this work and what would be gained? Taking a PiP user as an example, an ontology engine might be used to prompt the user with a set of possible communities that the ontology recognises and which could be achieved with the currently available devices. This would help a novice user to setup an acceptable world with a minimum level of intervention on their part. The options offered might be graded, either all possibilities, or options related to high-level functions described by the user with perhaps, the high-level requirement itself captured as an ontology. This process might be implemented as a 'virtual helper' suggesting the range of possible virtual devices that could be built from the currently available devices. As all technology has to have commercial potential, an opening for this might be that the ontology engine could suggest devices the user might consider buying. The ontology descriptions themselves (MAps) would also have a commercial value and open up the possibility for new forms of trading in virtual commodities. More speculatively, if the system included an agent based learning mechanism, over and above any end-user programming, there might be patterns of use and behaviour of the sorts of (implicit) communities formed and their use, that could be captured by an agent from the (implicit) rules created, which in their turn, could be used to improve the advice offered by the helper system. In terms of the underlying science, such an approach could unify implicit autonomous agent learning mechanisms with explicit end-user programming. In terms of the levels of abstraction involved one might characterise this as an epistemological level as it seems to capture tacit knowledge from the behaviour of the user rather than rely on what the user knows and wants consciously and explicitly. There would be a degree of recursion in this process as epistemological level processes could result in ontological instantiations, which in turn could feed the epistemological level modelling potentially leading the user to possibilities nobody had considered. The point being that through PiP, the user's beliefs and desires are captured at an epistemological level which via an ontology could add a more personal aspect to the prompts and suggestions offered to the user. Of course, using PiP, the user could still invent new virtual devices which could then become part of an expanded (personal) ontology however well or badly they were formed. In some senses both the ontology and the epistemology described here are dynamic bodies of knowledge and belief that evolve as the pervasive system and the users evolve. The basic ontology of devices would probably be manufacturer based and refer to physical device descriptions and capabilities whereas the virtual devices constructed by the user would be the equivalent of an epistemological level (i.e. what the user wants and knows how to construct for their own purposes). Clearly, there is equivalence between the epistemological level and a personal ontology creating the potential to encapsulate personal and subjective views which are especially in keeping with the domestic pervasive information systems in the private spaces of our homes.

Acknowledgements:
We wish to record our gratitude to the following who, in various ways, have supported this work: The DTI (Next Wave Technologies and Markets programme), Chimera (institute for socio-technical research), Essex University (our home university) and especially our colleagues Dr Martin Colley, Dr Hani Hagras, Dr Martin Hicks, Greg Willat and Malcolm Lear.

References:

Adair M. “X10 Projects for Creating a Smart Home”, *Indy-Tech Publishing*, ISBN: 0790613069, March 2005.

Andersen P.; Bardram J,; Christensen H,; Corry A.; Greenwood D.; Hansen K.: Schmid R. Open Architecture for Palpable Computing Some Thoughts on Object Technology, Palpable Computing, and Architectures for Ambient Computing, Discussion Document on *Palpable Computing architecture* (see http://www.ist-palcom.org/ accessed on the 15th October 2005).

Asaravala A. Give Your Home a Brain for Xmas, *Wired*, Dec 2004.Le Corbusier. Towards a New Architecture (Vers Une Architecture)”, *Architectural Press*, 1921 (reprinted 1989).

Basu J, Callaghan V “*Towards A Trust Based Approach to Security and User Confidence in Pervasive Computing Systems*”, IE05, Essex, 28-29th June 05

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

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

Canfield-Smith D.; Cypher A.; Tesler L. Programming by example: novice programming comes of age, *Communications of the ACM*, Volume 43, Issue 3 (March 2000), Pages: 75 – 81, 2000.

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

Chin J.; Callaghan V.; Colley M.; Hagras H.; Clarke G. Virtual Appliances for Pervasive Computing: A Deconstructionist, Ontology based, Programming- By-Example Approach, *Proceedings of Intelligent Environments 2005 (IE05)*, Essex, 28-29th June 05.

Chin J,; Callaghan V,; Colley M,; Hagras H,; Clarke G. End-User Programming in Pervasive Computing Environments, *Pervasive Systems and Computing (PSC-05)*, Las Vegas, Nevada, USA, June 27-30, 2005.

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

DCOMP. *Deconstruction and Community Based Ontology For Pervasive Computing* (http://iieg.essex.ac.uk/dcomp/ont/dev/2004/05/ accessed 15th October 2005)

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

DTI. Next Wave Markets, *UK Department of Trade and Industry web pages*; (available at www.nextwave.org.uk/docs/markets.htm, accessed 15 Oct 05).

Easy Living. Microsoft’s Easy Living Project (available at http://research.microsoft.com/easyliving/ accessed on the 15th October 2005).

Economist. The digital home; Science fiction? *The Economist*, Thursday September 15th 2005.

Epsilon. SUN's Epsilon Project (available at http://research.sun.com/projects/epsilon/) accessed on 15th October 2005.

Haarslev V and Moller R. Description of the RACER system & its application, *Proc Int'l Workshop on Description Logics* (DL-2001), 2001.

Horan B. The Use of Capability Descriptions in a Wireless Transducer Network, Sun *Microsystems Research Labs, Report Number: TR-2005-131*, Feb 1, 2005 (available at http://research.sun.com/techrep/2005/abstract-131.html, accessed 15th October 2005).

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

Jobs S. Tiger (Mac OS X), *Apple Worldwide Developer Conference WWDC 2004*, San-Francisco, 27 June – 3rd July 2005 (available at http://developer.apple.com/macosx/automator.html, accessed 15th October 2005).

Le Corbusier "Towards a New Architecture**", Architectural** *Press*, 1970

Lieberman H. *Your wish is my command*, Morgan Kaufmann Press, 2001

Lyons M. "Privacy, Freedom and Control in Intelligent Environments" keynote talk, *Proceedings of Intelligent Environments 2005 (IE05)*, Essex, 28-29th June 05, (http://www.iee.org/OnComms/PN/controlauto/Michael%20Lyons%20Presentation.pdf accessed 15th October 2005.

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.

McBride B. Jena: A Semantic Web Toolkit, *IEEE Internet Computing*, Nov/Dec 2002

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.

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

OWL-S. *Ontology Web Language for services*, (http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/ accessed 15th October 2005)

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

Wacks K. Home Automation and Utility Customer Services, *Cutter Information Corporation Report*, April 1998

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

Zamudio, V.; Callaghan, V.; Chin, J. A Multi-Dimensional Model for Task Representation and Allocation in Intelligent Environments, *Proceedings of The Second International Symposium on Ubiquitous Intelligence and Smart Worlds* (UISW2005). Nagasaki, Japan. 6-7th December, 2005.

Zou Y,; Chen H,; Finin T. F-OWL: an Inference Engine for Semantic Web, *3rd NASA-Goddard/IEEE Workshop Formal Approaches to Agent-Based Systems*, Greenbelt, Maryland, USA, 26 April 2004.