Pers Ubiquit Comput (2004) 8: 255–263
DOI 10.1007/s00779-004-0286-7

ORIGINAL ARTICLE

Irene Mavrommati · Achilles Kameas
Panos Markopoulos

An editing tool that manages device associations in an in-home environment

Received: 15 February 2004 / Accepted: 9 April 2004 / Published online: 26 June 2004

Abstract The forthcoming home environment will comprise numerous computationally enhanced artifacts that are autonomous, but interconnected via an invisible web of network-based services. The approach presented in this paper is to enable end users to make their own applications by linking such artifacts, which are treated as reusable "components." A key requirement to achieve this is the availability of editing tools that meet the needs of different classes of users. A tool of this kind designed for end users is presented in this paper, together with the outcome of user evaluation sessions.

Keywords Ubiquitous computing · Component architectures · End-user programming

1 Introduction

A number of information appliances and "smart" products from the consumer electronics and white goods industries are gradually being introduced into consumers' homes. Mundane objects of every-day use that are enhanced with computing and communication capabilities are being developed experimentally, whilst some are appearing on the market already. Some examples of such processing- and communication-enabled products are the Internet fridge, Internet microwave oven, UPnP hi-fi sets and DVD players, sensor-enabled toys, such as Furby and other pet or robotic toys, tagged coffee mugs, digital picture frames, sensor-/Internet-enabled furniture, to mention but some. It seems that our future environments will consist of an increasing number of objects, furniture, and appliances that will be enhanced with information communication capabilities, which will be able to work synergistically with each other [1]. The future home is expected to become populated by distinct devices that are interconnected via an invisible web of network-based services [2, 3]. The research challenges relating to creating this future home environment are not only technological (i.e., how do the objects and appliances share common standards, how they communicate robustly with each other). Equally important are the challenges that relate to the acceptance of this new metaphor by people, and the resulting human behavior (how do people develop an affinity with these environments; how will the mental models of their environments be re-shaped; what are the skills they will develop to cope with behavioral dependencies between objects that are not always visible) [4].

Several research projects are currently underway, aiming to develop the necessary hardware and software modules that will enable devices to communicate and collaborate [5, 6, 7]; thus, we can safely assume that future in-home devices will be functionally autonomous, but able to synergize with each other, regardless of the differences in their shape and functionality [4]. In project e-Gadgets [8] we consider them as components of the in-home environment, which can be freely associated with several different ways, thus, collectively achieving different functions within the home. The functionality of collections of objects may serve different purposes: pleasure, play and fun, or home automation and task facilitation.

The ability to "configure" and "reconfigure" [6] the in-home devices stands as a driving concept behind this vision, as it allows for end-user creativity to emerge in a ubiquitous environment, where people can create their

I. Mavrommati
Department of Products and Systems Design Engineering, University of the Aegean, Greece

I. Mavrommati (✉) · A. Kameas
Computer Technology Institute, R. Ferraiou 61, 26221 Patra, Greece
E-mail: mavrommati@cti.gr
Tel.: +30-2610-273496
Fax: +30-2610-222086
E-mail: kameas@cti.gr

P. Markopoulos
Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: P.Markopoulos@tue.nl

own niche applications or adapt their ubiquitous surroundings [4, 9]. In order to realize this possibility, firstly, one has to develop the concepts that will serve as a common referent among people, designers of applications, and developers of technology. Based on these, one has to:

1. Implement a universally applicable model of the communication between components of the system (that is, services, distinct objects, and in-home appliances). Such a model needs to provide a common technology-independent upper layer (providing the end-user functionality) and a lower layer that can accommodate existing standards, system architectures (such as UPnP, Jini, etc) and communication protocols (such as Wi-Fi, Bluetooth, etc.).
2. Build mechanisms (appropriate tools/interfaces) with which people can act upon their environment, and manipulate the processing appliances, objects, and services within it. Such control mechanisms can be associated with existing appliances or built anew.
3. Reinforce the willingness and the ability of people to use the in-home environment by developing tools that provide them with a basic level of understanding of the in-home-environment and the applications running. Such an environment will consist of tangible or intangible components invisibly interconnected via local or remote network links. Thus, the tools must visualize the structure of the environment and applications, present their current state and content, explain their functionality, and enable (or help) people manage the inter-component associations.

In e-Gadgets, we defined and refined the Gadgetware Architectural Style (GAS), which includes a set of concepts and application rules, an enabling middleware, a methodology, and a set of tools that enable people to compose distributed applications from the services offered by artifacts and devices. GAS applies concepts of component-based software engineering to ubiquitous computing environments [10].

In this paper, we present a control device via which a number of in-home devices can be associated in different ways: an *editor* which allows a greater control over the in-home environment. We shall not delve into the particulars of GAS or other ubiquitous computing architecture/middleware, but, for the scope of this paper, we consider that the interoperability between objects is given, as it can be realized by existing/proposed architectures. After presenting the design and functionality of the editor, we shall discuss the outcomes of a concept test conducted with users and we shall give an indication of people's reactions to the editor.

Ubiquitous computing applications will have a widespread impact in the home only if people can understand a basic set of underlying technology concepts behind ubiquitous computing; they may, thus, develop a feeling of trust by experiencing their ability to control such systems. As a consequence, in our research, we did not adopt a 'black-box' engineering approach where users are not able to observe the structure of the system they are experiencing, but rather, we used an opaque approach which partially discloses the structure of a system. In the end-user-trials, the validity of an ''opaque'' approach for ubiquitous computing engineering is assessed. The same concepts and constructs are provided in a more detailed level to the manufacturers and builders of the applications, and may also be used by intelligent mechanisms to adapt them in a black-box-type approach. The approach of adopting an editor mechanism towards the (re)configuration of ubiquitous computing applications by people aspires to provide them with a greater degree of transparency into the ubiquitous computing home.

2 Devices as components in the home

One assumption underlying GAS is that appliances and objects in the home are manufactured to be autonomous and functionally self-contained. Moreover, they can locally manage their resources (processor, memory, sensors/actuators, etc.). GAS provides a compatibility framework that is a layered middleware that enables them to share definitions of services, exchange data, interpret incoming messages correctly, and act accordingly. Thus, GAS enables in-home artifacts and appliances to be used as components of a non-a-priori-defined system, and, at the same time, enables users to play an active role in understanding and defining the functionality of their ubiquitous environment. For this, GAS provides a middleware that can directly interface with hardware components and also serves as a layer on top of existing network protocols or distributed system architectures, enhancing them with GAS-specific functionality.

The objects in an in-home environment can range from simple (tags, lights, switches, cups) to complex ones (PDAs, hi-fi, fridges), and from small ones (sensors, pens, keys, books) to large ones (desks, chairs, carpets, rooms). By associating these home devices together into collaborating collections, people can shape their own environment. Such an approach supports the development of open systems [11] and, at the same time, can be used to explain the system to people; this is necessary in order for them to be able to manipulate such environments.

At the conceptual level, GAS includes the *plug–synapse model*, which regards each object in an in-home environment as having a set of abilities and offering or requesting a set of services; these abilities, which can be inter-associated, are visualized via the software construction of *plugs*. People can associate compatible plugs (thus, creating *synapses*) and establish a composition of the respective services/functions (Fig. 1).

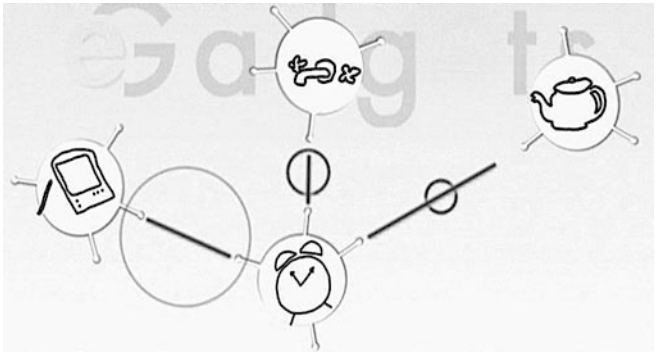

Fig. 1 An abstract visualization of the plug–synapse model

2.1 An example

Let's take a look at the life of Patricia, a 27-year-old single woman, who lives in a small apartment near the city center and studies Spanish literature at the Open University. A few days ago, she passed by a store where she saw an advertisement about "extrovert Gadgets." Pat decided to enter. Half an hour later, she had given herself a very unusual present: a few furniture pieces and other devices that would turn her apartment into a smart one! The next day, she was anxiously waiting for the delivery of an e-Desk (it could sense light intensity, temperature, and weight on it), an e-Chair (it could tell whether someone was sitting on it), a couple of e-Lamps (one could remotely turn them on and off), some e-Book tags (they could be attached to a book, tell whether a book is open or closed, and determine the amount of light that falls on the book), and... an e-Carpet (you just had to step on it). Pat had asked the store employee to pre-configure some of the e-Gadgets so that she could create a smart studying corner in her living room. Her idea was simple (she felt a little silly when she spoke to the employee about it): when she sat on the chair and drew it near the desk and then open a book on it, the study lamp should be switched on automatically. If she closes the book or stands up, then the light should go off (she hadn't thought of any use for the carpet, but she liked the colors).

For research purposes, a number of domestic objects and furniture have been augmented with computation and communication capabilities and turned into e-Gadgets. This "GASification process" is a stepwise methodology that ensures that all the necessary hardware and software modules are installed in the object and initialized correctly.

In order to turn an everyday object into an e-Gadget, firstly, one has to attach to it a set of sensors and actuators. For example, in order for the e-Table to be able to sense weight, luminosity, temperature, and proximity, it has to be equipped with pressure pads, luminosity sensors, and an infrared sensor. Pressure pads are mounted underneath its top surface and cover its entirety. Luminosity and temperature sensors are evenly distributed on the surface. Four infrared sensors are placed on the legs of the table and another on its top (Fig. 2).

There's a lot of circuitry that connects these sensors with power sources (replaceable batteries in this case) and with driving hardware modules. The latter are required in order to collect and filter sensor readings into the GAS-OS middleware, which currently runs on an iPaq attached to the object (to preserve autonomy of objects, an iPaq per artifact was used, providing the required processing and communication hardware).

Once the hardware is installed, one has to install GAS-OS in the artifact's iPaq and configure it to interface with the hardware. This is achieved via a special software module, the Gadget-OS, which interfaces with an FPGA that drives the sensors. When these objects have reached beyond the prototype stage, all necessary hardware (including sensors, processor, wireless module, battery, boards, and circuitry) will be embedded into them during their manufacture; all that will be required is to download GAS-OS in them and configure it for the specific object.

The latter step includes the definition of the artifact's ID and its plugs (properties, services, capabilities) in a way that will be understandable by other artifacts. This is done by creating XML-based descriptions of plugs using universally agreed core ontology (a vocabulary of basic terms) [12]. The uniqueness of the ID is achieved with a process similar to the one used for Mac addresses [13]. All these are used by the GAS-OS modules in order to manage plugs and synapses, create sensor readings into messages, translate incoming messages into service requests, etc. GAS-OS runs in Java, but relies only on features available in Java Personal Edition, compatible with the J2ME personal profile. This allows deployment on a wide range of devices, from mobile phones and

Fig. 2 The top surface of the e-Table and the supporting circuitry underneath

PDAs to specialized Java processors. The proliferation of end-systems capable of executing Java make Java a suitable underlying layer, providing a uniform abstraction for the middleware.

3 The editor functionality

3.1 The editor's interaction model

People can supervise the functional capabilities of devices in the home via other specialized devices; the editors. GAS editors currently provide an integrated graphical user interface (GUI); potentially, they may be linked to multimodal interfaces integrated in the environment, such as speech or gesture input systems. With editors, it is possible to supervise and interface with the various in-home appliances. One can supervise the interconnectable capabilities of an appliance (the plugs). Finally, people can act upon these capabilities by associating them into matching pairs that serve specific functions (the synapses); they can break existing associations, pause them, or add parameters in the association to influence the details of the function (Fig. 3). The synaptic associations created/edited are not stored locally (i.e., in an editor), but are stored in the distributed appliances and objects within the home environment. Thus, in the case of failure or object movement beyond network range, much of the application functionality can be restored.

The editing capabilities can be used by designers to create ubiquitous computing applications, without having to start from scratch, as they may reuse existing component objects [6]. They may be used by end users to personalize ubiquitous applications, or for exercising their own creativity in building novel associations for niche or innovative functions. The core editor functionality can also be accessed directly by intelligent agents that construct and adapt applications by monitoring user behavior.

3.2 Examples of linking appliances as components

The behavior requested by Pat requires the following set of e-Gadgets: e-Desk, e-Chair, e-Lamp, e-Book. The collective function of this application (we use the term Gadgetworld for GAS-compatible ubiquitous computing applications) can be described as: when the particular CHAIR is NEAR the DESK AND ANY BOOK is ON the DESK, AND SOMEONE is sitting on the CHAIR AND the BOOK is OPEN THEN TURN the LAMP ON.

In order to achieve the collective functionality required by Pat, the employee in the store had to create a set of synapses among e-Gadgets' plugs (Fig. 4). This type of functionality and component structure is created, inspected, and modified through the editor. For example, Pat can subsequently define the intensity of the e-Lamp when it is being automatically switched on; thus, the light won't blind her. Or, if an intelligent agent is used, it could adjust the light intensity each time, based on the overall amount of light in the room, as it is recorded by luminosity sensors distributed on objects in the room.

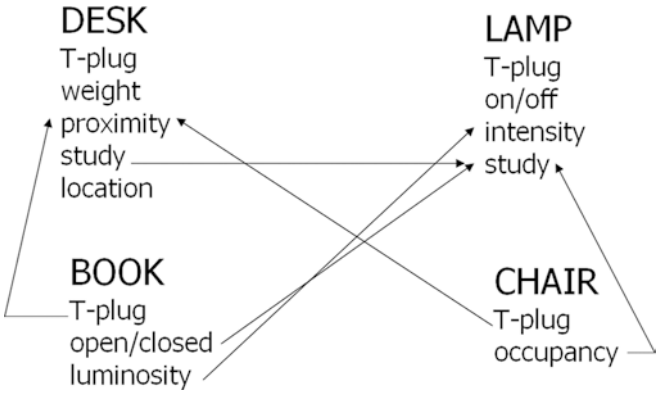

Fig. 4 Schematic representation of the connections between appliances in the described scenario

Fig. 3 Use of the PDA-based editor by a test subject

3.3 Editing functionality

The main role of the editor is to allow the user to create and edit synaptic associations between appliances. An association can be established between two capabilities when these are available via the digital self of a device. The user has to indicate which two capability plugs to associate and link them with each other; therefore, a cause–effect relationship is forged between the objects. The editor uses the services of GAS-OS to support editing actions.

The purpose of the editor is threefold:

1. To indicate/make visible the available enhanced devices and appliances in one's home environment, and their existing functional dependencies
2. To form new associations between devices in order to achieve certain functions or to insert new devices and objects in a Gadgetworld
3. To assist with debugging, editing, servicing, etc.

The editor identifies the information communication devices in the vicinity of the home. It also sees the capabilities offered by each device that can be interconnectable. Such capabilities have a direct relationship with the actuating/sensing capabilities of the objects and the functions that are intended by the appliance manufacturers. Nevertheless, some of these capabilities (especially more complex ones) may not be obvious to people, apart from via the editor.

In addition the editor identifies the current working groups of associated appliances in the home (the current configurations that are available) and displays them for supervision. The links between the compatible capabilities of appliances are visualized and can be manipulated through the editor. In the test bed implementation, this is done by a GUI using an association matrix, as shown in Fig. 7. Associations between certain capabilities of appliances/objects can be formed, thus, creating configuration sets for a certain purpose.

The identification and selection of capabilities (plugs) via the editor is a task that depends on the user expertise. A novice user might not be interested in, or understand, more than just the description of the capability; (s)he can then base his/her selection on a natural language description that is proposed in the design of the editor. A more advanced/experienced user may prefer to see more of the technical details in order to make his/her selection of the capability.

Once such a set of synaptic associations is established, the part of the operating system that runs on each participating device ensures that it works. The associations that are activated will operate as long as the user wants them to (until (s)he deactivates or deletes them), unless there is a technical inability to maintain its functionality (i.e., one participating object is out of range, non-responsive, etc.).

3.4 High-level architecture

The editor is designed to be compatible with the in-home appliances. It is a piece of software that can run on top of an existing information appliance (such as a PDA or a PC), while the top software layer utilizes the particular interface resources of the appliance it is residing at. Therefore, its core is independent from the device it runs on, allowing for a multitude of interaction modalities to be implemented. Two implementations have been created using on-screen interfaces, on a handheld computer and a laptop computer (Fig. 5). This enabled us to test the editor concepts in a portable form.

The editor's high-level architecture consists of three layers (Fig. 6):

- The operating system layer, which offers to the editor its communication capabilities and knowledge of other connectable appliances interfaces
- The editor manager layer, which provides abstraction of the interface layer, as well as compatibility with the function layer. It contains all the structures and functionalities needed by the editor.
- The user interface layer that is responsible for providing any interactions with the end user, as well as any other operations that the editor provides

3.5 Current editor implementation

Two versions of the editor have been created in order to test the primary editing functions. The one with richer functionality runs on a laptop or PC and was designed with "professional" designers in mind. The second, and

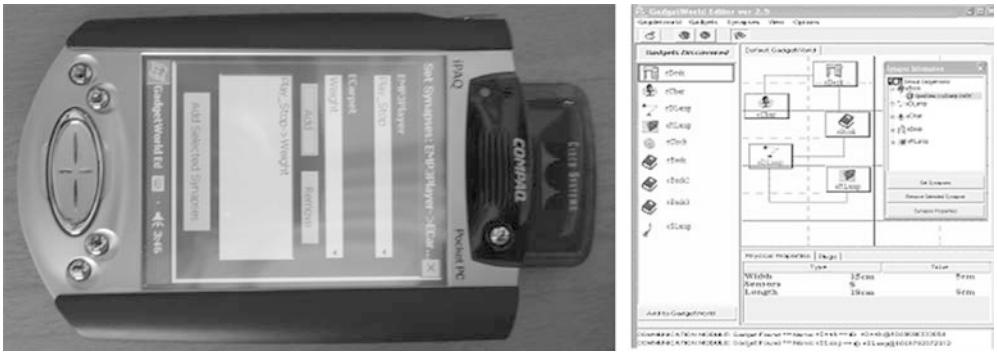

Fig. 5 Two implemented versions of the editor on PDA (*left*) and PC (*right*)

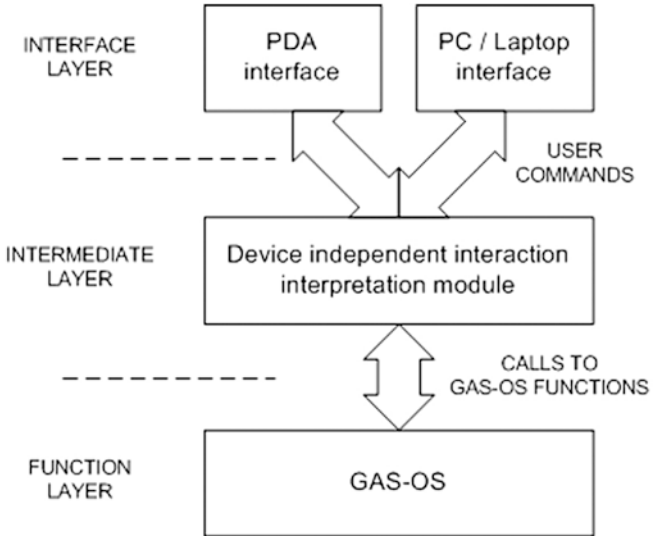

Fig. 6 Schematic representation of the editor layers

simpler one, runs on an iPAQ handheld computer and is intended for the non-trained end user. Twelve sample devices have been modified in order to be compatible so that they can be seen and associated in a number of ways by the editor. In the current implementation, the GUI is handled at the interface layer and all the functions required of the GUI (like discovery of devices, activation of functional association sets, etc.) are mapped to the actual operating system functions in the function layer, via the intermediate layer.

The editor designed for use on a laptop or PC consists of several panes. There is a pane that lists the devices in the vicinity, called the *listing* pane. From the listing pane, the user can drag-and-drop selected devices into the *editing* pane. Upon selection of two devices, an association matrix that shows both devices' capabilities opens up. Tagging a square on this matrix associates the two capabilities (with preset properties). If the user wants to change the specific details of this association, they can do so by opening the *mappings* window, and modifying the properties of each capability in the association. The newly created application then needs to be named and activated. When this has been done, the user can test the association set by using the physical devices (in order to ensure that the configuration is working as planned); alternatively (s)he can use the editor and adjust the association mappings (properties) or delete the total set of connections.

The PC interface of the editor (Figs. 4 and 7) allows several ways of working—the two most prominent are the clockwise operation and the counterclockwise operation. In the clockwise operation, the user creates a new configuration starting from the selection of the devices; then, (s)he identifies which capabilities (s)he wants to use and links them in an association and finally specifies the parameters of this association (its properties). The counterclockwise operation is suggested so that the formation of the Functional set can be based on a description of the functionality that the user intended. (Searching for already established sets (based on similar target functions) and using a natural language description interface in order to figure out the devices needed and the synapses required to achieve the requested function).

The panes that are suggested in the design of the editor allow the editing task to start from any level: one can search for the capabilities required first and then deduce the devices, or one can search for devices based on their class and perform operations on sets of the devices.

In the working prototype of the editor, the primary functions that enable the creation of associations were quite robustly implemented (discovery and visualization of artifacts and their capabilities; creating associations; setting the properties of these associations, naming and activating a functional set of associations; activating, deactivating, deleting each association or each set), while the auxiliary natural dialog and facilitating operations part have not been implemented.

The working prototype using the screen and GUI on a PC/laptop gives the primary functions using similar panes as the ones suggested in the design phase. It provides in a rather extended way the editing items for supervision and manipulation.

The working prototype version of the editor on a PDA contains the same primary functions, but in a more condensed way that guides the user through a stepwise

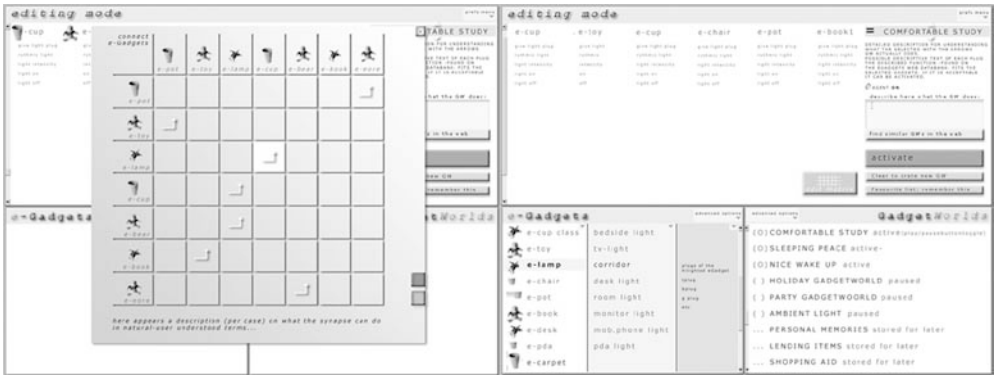

Fig. 7 Draft design of a GUI proposed for the editor (for a PC)

process for creating a Gadgetworld. This is partly dictated by the small display size. Due to its stepwise approach and the limit imposed on the amount of presented information and options at any one time, it is easier to use. Consequently, the PDA-based implementation was the one used for evaluation purposes (Fig. 3).

4 Concept evaluation

A central research question is how acceptable it is for end users to manipulate the various devices in the home environment. This is linked to the extent to which users comprehend the concepts underlying the editor, and their willingness to use it. A concept evaluation was conducted with the editor that revolved around two axes: comprehensibility of the underlying concepts and willingness to use such technology. The editor was evaluated as one part of a broader system and concepts that were proposed by the e-Gadgets project. The editor evaluation did not focus on the specific design details, i.e., the look and feel of the GUI. Rather, it aimed to assess the role that "editing" can play in the future home environment and how it the extent to which it is perceived to satisfy user needs.

Initially, an expert appraisal was carried out to evaluate the proposed interaction concepts and technology with respect to the end-user requirements. The evaluation was conducted in phases. First, an expert review was conducted in the form of a workshop. Subsequently, the cognitive dimensions framework [14] was applied to assess how well the e-Gadgets concepts support end users in composing and personalizing their own ubiquitous computing environments. In this first round of evaluation, a first version of the editor was used, together with paper mock-ups. The nature of the evaluation was formative, i.e., it aimed to suggest directions for the next steps of the project which would ensure that user needs are taken into account. After a working prototype of the editor was developed (using a GUI in a PDA), the concepts were tested through a hands-on demonstrator at two events and a wealth of questionnaires were collected.

4.1 Expert appraisal

The end user that would engage in editing applications in one's in-home environment is considered to be a "technophile" but not a programmer; three experts in user system interaction that matched this profile performed a set of evaluation activities during a workshop evaluation session (the workshop activities and outcomes are described extensively in [15]). The basic concepts were introduced in a short introductory session to familiarize experts. Four scenarios were discussed which highlighted different usage/interaction design issues. A small discussion in a focus-group-format followed each scenario. A problem solving exercise was set to gauge the extent to which these experts could construct an in-home application and, further, to reflect on what they consider as problems of doing this. A design draft (paper mock-up) of the GUI proposal for the editor and some video prototypes presenting different multimodalities were demonstrated and expert opinions were solicited. Finally, an open-ended discussion elicited global level feedback for the project.

In the evaluation session, a broader set of issues was addressed, relating to a model of a component-based approach for the ubiquitous computing home. A subset of the evaluation tasks concerned the editor. Some of the most recurring themes of the discussion relating specifically to the editor were:

- Ubiquitous computing technologies embedded in physical objects add hidden behavior and complexity to them. Problems may arise if this behavior is not observable, inspectable, and predictable for the user.
- Intelligence causes problems of operability and unpredictability for users. It must be used with caution and this should be reflected in the demonstrations built.
- Constructing and modifying applications in the ubiquitous home is a problem solving activity performed by end users. As such, it has an algorithmic nature and, thus, good programming support should be offered.

4.2 Cognitive dimensions evaluation

Following the last observation, the expert conducting the evaluation has conducted a further assessment using the cognitive dimensions framework [14]; a broad-brush technique for the evaluation of visual notations or interactive devices. It helps expose trade-offs that are made in the design of such notations, with respect to the ability of humans to translate their intentions to sequences of actions (usually implemented as programs) and to manage and comprehend the programs they compose. Broadly, the editor facilitates the creation of in-home applications that are composed in non-textual manner. Thus, this theoretically founded technique can be used to provide insight into selecting between alternative choices with respect to providing tools for applications construction. Some of the most interesting points resulting from the evaluation with the cognitive dimensions framework were the following:

- There will always be an initial gap between their intentions and the resulting functionality of a user-composed application. Users will have to bridge this gap based on the experience they develop after a trial-and-error process. An editor can shorten this initial gap by allowing several different ways of expressing the user's goals.
- Since an object can be part of several in-home applications at the same time, the effect it has on each is not easy to understand from the physical appearance.

Developments of the editor will need a way to illustrate to the user how the specification of the parts influences the dynamic behavior of the whole application (similar to debuggers in object-oriented environments).
- Editors should aim to bridge the gap between architectural descriptions of an application and the user's own conceptualizations, which might be rule-based, task-oriented, etc. (improving the closeness of mapping dimension).
- To edit in-home applications, the editor requires only a few conventions to be learnt by the end user (appropriately low terseness).
- The editor should make observable logical dependencies between seemingly unrelated physical objects (hidden dependencies). There are side effects in constructing applications. A state change in one component may have non-visible implications on the function of another. In the conceptual diagrams used during the discussion (Fig. 3), dependencies were directly visible. However, in the GUI mock-up shown in the evaluation, connections and their rules are not shown. Some way of visualizing and inspecting such connections needs to be added.
- An object can belong to several applications; the effect it has on each is not easy to understand from the physical appearance (role expressiveness). (A way to show this is to represent it via the editor.)
- The abstraction level is appropriate for the target user audience (abstraction gradient dimension).

4.3 Surveys at two conferences

An the second stage, the working editor prototype developed (using a GUI of a PDA and one on a PC) was presented through two hands-on demonstrations at two events. One session was during the "Tales of the Disappearing Computer" event in May 2003 (where ten completed survey questionnaires were received). The second session was at the British Annual Conference on Human Computer Interaction (where 29 completed questionnaires were received).

The demonstration featured an editor running on a PDA that supported discovery and use of another three appropriately converted devices in the room: a Mathmos tumbler light, an MP3 player, and a pressure-sensitive floor mat. Conference delegates were invited to make their own associations by connecting the components available using the editor. After a short introduction, delegates were able to compose applications in order to control the music played by positioning themselves on the floor mat or by flipping the Mathmos tumbler on its side (as it resembles a luminous brick).

A wealth of comments was collected in the questionnaires. For example, thirteen delegates found that this technology will not be used because it is too complex; eleven noted that is very easy to create and modify applications, and four said that it is very easy to learn to use it. Five delegates noted that it would not be easy for users to appreciate the benefits of this technology.

4.4 Sum up of the evaluation outcomes

The feedback that relates to the editing tools can be summed up as follows:

- The application behavior should not surprise the user, i.e., automation or adaptation actions should be visible and predictable (or at least justifiable).
- End users acting as ubiquitous application developers should be supported with at least as good tools as programmers have at their disposal, e.g., debuggers, object browsers, help, etc.
- Multiple means to define user intentions should be supported by the GUI of the editor, as the users' tasks tend to be comprehended and expressed in a variety of ways.
- The acceptability of the Gadgetworld concepts depend on the quality of the actual tools and their design.

5 Conclusions

We attempted to provide a level of transparency into the ubiquitous environment by giving end users the ability to construct or modify ubiquitous computing environments. To do this, concepts of component-based software were taken and applied in order to treat physical objects as components of ubiquitous computing environments.

A novelty of the editing approach in the home environment is that artifacts are treated as reusable "components." The component architecture is made directly visible and accessible via the editor. This enables end users to act as programmers. This end-user programming approach may be especially suitable for ubiquitous computing applications. The possibility to reuse devices for several purposes—not all accounted for during their design—opens up possibilities for emergent uses of ubiquitous devices whereby the emergence results from actual use.

As a lot depends on the editing tools in terms of interface and interaction, and also extensive auxiliary functionality to aid with the editing tasks, our future work aims to address the GUI design and development of one complete editor in an iterative design process.

Acknowledgements We would like to thank N. Drossos and J. Calemis for developing the working software of the editor.

References

1. Aarts E, Marzano S (eds) (2003) The new everyday: visions of ambient intelligence. 010 Publishing, Rotterdam, The Netherlands

2. Aarts E, Harwig H, Schuurmans M (2001) Ambient intelligence. In: Denning JP (ed) The invisible future. McGraw Hill, New York, pp 235–250
3. Harwig R, Emile A (2002) Ambient intelligence: invisible electronics emerging. In: Proceedings of the international interconnect technology conference (IITC 2002), San Francisco, California, June 2002
4. Mavrommati I, Kameas A (2003) The evolution of objects into hyper-objects: will it be mostly harmless? In: Proceedings of the 1st international conference on appliance design (1AD), Bristol, UK, 6–8 May 2003
5. The Disappearing Computer (DC) initiative, http://www.disappearing-computer.net/
6. Newman W, Sedivy J, Neuwirth CM, Edwards K, Hong JI, Izadi S, Marcelo K, Smith TF (2002) Designing for serendipity: supporting end-user configuration of ubiquitous computing environments. In: Proceedings of the conference on designing interactive systems (DIS 2002), London, UK, June 2002, ACM Press, pp 147–156
7. Humble J, Crabtree A, Hemmings T, Åkesson KP, Koleva B, Rodden T, Hansson P (2003) "Playing with the bits": user-configuration of ubiquitous domestic environments. In: Proceedings of the 5th international conference on ubiquitous computing (Ubicomp 2003), Seattle, Washington, October 2003
8. Extrovert Gadgets (e-Gadgets) Projects, http://www.extrovert-gadgets.net
9. Rodden T, Benford S (2003) The evolution of buildings and implications for the design of ubiquitous domestic environments. In: Proceedings of the CHI 2003 conference on human factors in computing, Florida, USA, 5–10 April 2003
10. Kameas A, Bellis S, Mavrommati I, Delanay K, Colley M, Pounds Cornish A (2003) An architecture that treats everyday objects as communicating tangible components. In: Proceedings of the 1st IEEE international conference on pervasive computing and communications (PerCom 2003), Forth Worth, Texas, 23–26 March 2003
11. Schneider JG, Nierstrasz O (1994) Components, scripts and glue. In: Hall J, Hall P (eds) Software architectures—advances and applications. Springer, Berlin Heidelberg New York, pp 13–25
12. Christopoulou E, Kameas A (2004) GAS ontology: an ontology for collaboration among ubiquitous computing devices. Int J Hum Comput St (to appear in Protégé special issue)
13. Ringas D, Kameas A, Mavrommati I, Wason P (2002) eComP: an architecture that supports P2P networking among ubiquitous computing devices. In: Proceedings of the 2nd IEEE international conference on peer to peer computing (P2P 2002), Linköping, Sweden, 5–7 September 2002
14. Green TRG, Petre M (1996) Usability analysis of visual programming environments: a cognitive dimensions framework. J Vis Lang Comput 7:131–174
15. Mavrommati I, Kameas A, Markopoulos P (2003) Visibility and accessibility of a component-based approach for ubiquitous computing applications: the e-Gadgets case. In: Proceedings of the 10th international conference on human–computer interaction (HCI International 2003), Crete, Greece, June 2003