THE CONCEPTS OF AN END-USER-ENABLING ARCHITECTURE FOR UBIQUITOUS COMPUTING

Irene Mavrommati, Achilles Kameas[1]

Abstract
This paper describes the set of concepts that underlie the "Plug-Synapse" model, which supports end-users in composing and configuring ubiquitous computing applications. The technology that implements the model is briefly presented; all have been developed in the course of research during the e-Gadgets (IST-FET) project.

1. Introduction

The vision of pervasive and ubiquitous computing promises that the environments where we work, relax or commute will be furnished with an increasing number of computationally augmented artifacts. These artifacts may be information appliances or just ordinary objects enhanced with computing and communication capabilities. Ubiquitous computing technology will need to be deployed and used in an immense range of different contexts, to fit seamlessly into the lifestyle and life-patterns of very different individuals and to do so without requiring those individuals to attend to technology instead of their own daily pursuits [11]. Consequently, the ubiquitous computing services and applications must be able to adapt to varying and changing situations and configurations, mainly determined by the environments where they will be deployed, but nevertheless unforeseeable by their designers and developers,. One potential solution is to enable users to configure, customize or even construct their ubiquitous computing applications [8]. This solution offers several benefits: (a) applications are adapted in the best possible way to users' own requirements; (b) applications can be incrementally improved by their very users; (c) users are able to design their own environments and interactive experiences and thus become shapers of their own environment instead of being mere consumers of technology. The first steps towards realizing this approach is the design of a set of concepts that is common both to designers and users, and the provision of architectures and tools to implement them.

The research project "extrovert-Gadgets (e-Gadgets)" (www.extrovert-gadgets.net) was conducted under the IST-FET-Disappearing Computer initiative and has provided a set of concepts, a use model, an architecture and prototypical implementations to support this endeavor. We will describe briefly the basic concepts and the plug-synapse model that were developed within this project. The research presented extends the notion of component-based software architectures to the world of physical objects, thereby transforming objects in peoples' everyday environment into autonomous artifacts, the eGadgets, which can be used as building blocks of larger systems. The computational environments

[1] DAISy Group, Research Academic Computer Technology Institute, R. Feraiou 61, 26221 Patras, Greece, {Mavrommati, Kameas}@cti.gr

formed by such artifacts are intended to be accessed directly and to be manipulated by untrained end-users.

2. Concepts and outcome

2.1. Concepts and model

An *eGadget* is an everyday physical object enhanced with sensing, acting, processing and communication abilities. An eGadget is a functionally autonomous object, capable of managing its own resources (including power, processor, memory, sensors etc) and of engaging in communication actions with other associated eGadgets.

A *GadgetWorld* is a dynamic functional configuration of associated eGadgets, which collaborate in order to realize a collective function *(Figure 1)*. GadgetWorlds are the equivalent of software applications consisting of interacting components, exhibit collective behaviors and appear as functionally unified entities. A GadgetWorld can be considered as a distributed, ubiquitous computing application.

The Gadgetware Architectural Style (GAS) constitutes a generic framework shared by both artefact designers and users for consistently describing, using and reasoning about GadgetWorlds. GAS defines the concepts and mechanisms that will allow people (eGadget users) to define and create GadgetWorlds out of eGadgets, and use them in a consistent and intuitive way. GAS is expressed in the Plug-Synapse model, a conceptual abstraction that enables uniform access to eGadget services / capabilities / properties and allows users to compose applications that realize a collective behavior in a high-level programming manner: users form Synapses by associating Plugs, thus composing GadgetWorlds!

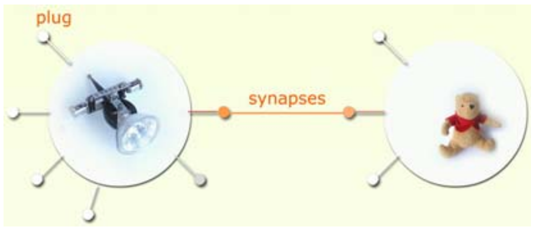

Figure 1: The plug-synapse model: eGadgets express their capabilities via the software construction of “plugs”. A “synapse” is shown as a connection between two “plugs”.

This approach provides a clear separation between computational and compositional aspects of an application, leaving only the second task to application designers and end-users. The benefit of this approach is that, to a large extent, system design is provided ready to the end user or the application designer, because the domain and system concepts are specified in the generic architecture.

Plugs are software classes that make the eGadget’s capabilities visible to people (through an editor) and to other eGadgets. Composition is effected through the definition of *Synapses* (links) between pairs of plugs. Each end of a synapse is managed by the GAS-OS running on each eGadget, thus implementing a peer-to-peer architecture. A synapse serves as the abstraction of a communication channel between peers. However, this occurs only when they have ‘discovered’ each other. Discovery may be forced upon the eGadget by a user creating the synapse with the editor, or proactively carried out by an eGadget (for example, when a light source breaks down, the switch connected to it may look for another light nearby, that can replace it). In addition, the option of an intelligent agent can be used to optimize GadgetWorlds, by learning from the ways people use them.

2.2. Example

As a simple but illustrative example *(Figure 2)*, consider the “Wake up Environment”, a GadgetWorld consisting of the following eGadgets: an alarm clock, a bed-mattress, the window blinds, and a room light. The eGadgets in this example have plugs through which they offer access to their properties and services. For example, the clock is equipped with a light sensor whose reading is made available through the plug “luminosity”; the bed-mattress has weight and pressure sensors and can decide whether there is someone “lying upon” or not; the window blinds offer the plug “open” which can be used to lift them to a specified height; finally, the room light offers the plug “on-off”. This GadgetWorld, clearly a ubiquitous computing application, can be set up to gradually increase the amount of light in the room (until the person gets up from bed) when its time for someone to wake up, first by opening the window blinds to let natural light come in, and, if it is still dark (i.e. consider a winter’s day when the lighting remains lower than a certain threshold), switch on the room light. The user has to synapse the “luminosity” plug of the alarm clock with “open” plug of the blinds and the “on-off” plug of the room light, synapse the “alarm on-off” plug with the “person lying” plug of the mattress (so that when the person stands up from the bed, the alarm is switched off and the blinds stop opening) and finally set the alarm clock. A second, simpler GadgetWorld could synapse the mattress with user’s slippers, and the latter with the coffee maker, thus starting coffee brewing when the person gets up and steps into the slippers. Moreover, an intelligent agent can be attached to one of the eGadgets and used to learn from monitoring user’s behavior in order to adapt the function of the GadgetWorld to better fit user’s preferences (i.e. stop coffee brewing when the user does not apply pressure in the slippers for more than two minutes, which might mean that the user has gone to the bathroom).

Figure 2: Snapshots of the e-Gadgets concept video, showing a few in-home application examples

2.3. Technology development

Plugs and Synpases constitute the logical architecture of a GadgetWorld; the system architecture that realizes it is shown in *Figure 3*. Note that in the general case, eGadgets form an ad-hoc network (hence the ability to support changing behavior), which becomes a P2P network when synapses are established.

GAS-OS [4] is the operating system / middleware that manages resources shared by eGadgets, determines their software interfaces and provides the underlying mechanisms that enable communication (interaction) among eGadgets. GAS-OS also provides synapse management, discovery and routing services at the GadgetWorld level.

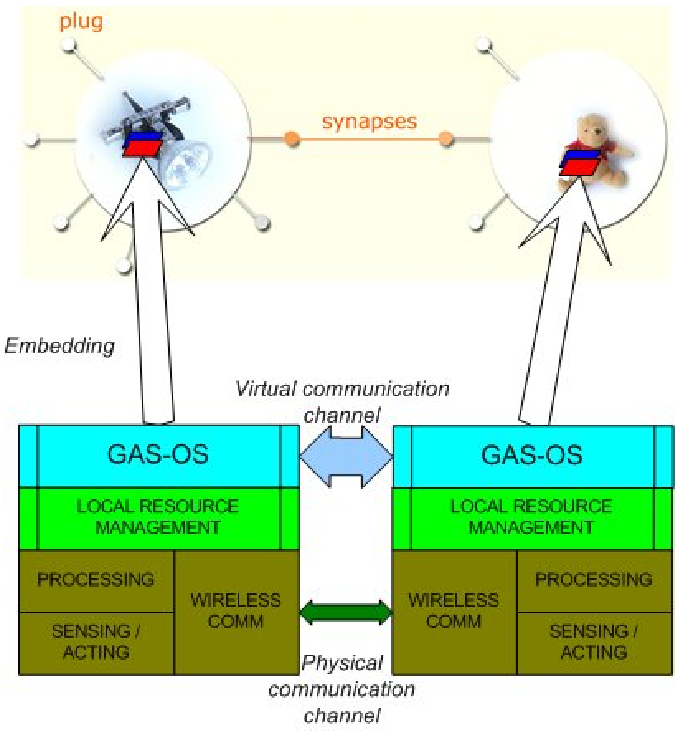

Figure 3: The logical and system architecture of a GadgetWorld

The current version of GAS-OS is written in Java. GAS-OS and supports IP-based communication using (without being bound to) IEEE802.11g. For the prototype implementation, we used iPAQ computers to execute it. A special board has been designed that includes the hardware required to interface GAS-OS with the sensors embedded in artifacts. GAS-OS supports the composition of e-Gadgets, without having to access any code that implements their interfaces.

Figure 4: A number of sample e-Gadgets running GAS OS have been created as test implementations

During the course of the project more that 12 sample eGadgets *(Figure 4)* have been created, as well as a student dormitory (the iDorm) at the University of Essex, as test implementations of embedding the proposed platform into everyday objects and settings. A software tool, the GadgetWorld Editor [5], was created to facilitate the composition of GadgetWorlds. The purpose of the Editor is threefold: a) to indicate/make visible the available eGadgets and GadgetWorlds b) to form new GadgetWorlds c) to assist with debugging, editing, servicing, etc. Two versions of the editor have been created. One with richer functionality runs on a laptop personal computer and has more detailed layers of functionality.

The second and simpler one runs on an iPAQ handheld computer *(Figure 5)* and is intended for the untrained end-user.

2.4. Evaluation

An expert review workshop, several demonstrations involving users [6] and a formal evaluation where test-subjects created and modified their own GadgetWorlds, were conducted with the working system, in order to evaluate the concepts put forth in the project. The e-Gadgets technology was demonstrated at two international events attracting experts in human-computer interaction. Around 30 delegates experienced the demonstration at the “TALES of the Disappearing Computer” (Santorini, Greece, May 2003) and 10 completed the feedback forms. Approximately 70 people visited the demonstration at the British HCI conference (Bath, September 2003) and we received 29 completed feedback questionnaires. The iDorm study was a combination of short tests and a single trial that took place overnight, aimed to get a relatively longer term and more realistic test of applied e-Gadgets technology. The combined outcome of the evaluation [7] indicated that the understandability of the Plug-Synapse model was high among users, and most were able to utilize the system and make simple configurations by using the concepts and tools provided.

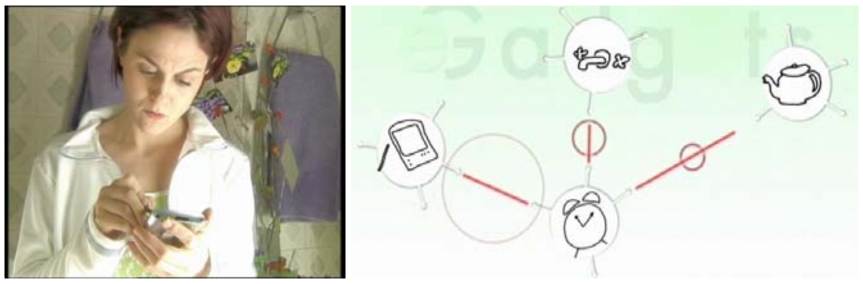

Figure 5: Use of the e-Gadgets editor, a tool that, together with the Plug-Synapse model, enables end-user programming of Ubiquitous Computing Applications

3. Related work

Several research projects are taking place in the broader area of ubiquitous computing systems. Most of them focus in specific, mostly technological parts of the architecture. In e-Gadgets, we attempted to provide a vertically integrated system that would offer a complete technological solution, while taking into account the user perspective. Among others:

- Gaia [9] provides an infrastructure to spontaneously connect devices offering or using registered services. Gaia-OS requires a specific system software infrastructure using CORBA objects, while mobile devices cannot operate autonomously without the infrastructure;
- BASE [1] and Universally Interoperable Core [10] represent component-oriented micro-kernel based middleware, which, although provides support for heterogeneity and a uniform abstraction of services, the application programming interface requires specific programming capabilities by users;
- TinyOS [3], an event driven operating system, designed to provide support for deeply embedded systems (i.e. sensor networks), which require concurrency intensive operations while constrained by minimal hardware resources;

- Commercial solutions (i.e. CORBA, JINI, SOAP) also exist, but are too heavyweight for the envisaged application domain and too complex to be used by people

4. Conclusions

The value of this approach is that, via the e-Gadgets tools, artefacts are treated as reusable components (for designers and people). Ubiquitous computing artifacts that follow the proposed architectural style can be reused for several purposes, in order to build a variety of Ubiquitous computing applications.

The proposed model is easily comprehendible; therefore by the appropriate use of tools, the e-Gadgets technology can be usable by designers of Ubiquitous computing systems, but also by untrained end-users. Subsequently this approach opens possibilities for emergent uses of ubiquitous artefacts whereby the emergence occurs from people's own use. Potentially it can enable the acceptability of Ubicomp technology into people's environments, as well as enable the making of emerging niche applications.

The solution we propose is communication (hence the term "extrovert"), as opposed to mere message exchange. Communication, according to Habermas, aims to achieve a shared understanding [2]. He says that one communicates in order to make known a desire or intention; then others can respond to the suggestion. In the e-Gadgets approach, a Synapse is formed as a result of negotiation among eGadgets. Negotiations and subsequent data exchange are based on ontologies possessed by the eGadgets; the only intrinsic feature of an eGadget is the ability to engage in structured interaction.

5. References

[1] BECKER, C., et al, BASE - A Micro-broker-based Middleware For Pervasive Computing, in Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication (PerCom03), Fort Worth, USA, 2003.

[2] HABERMAS, J., The theory of communicative action I and II (trans. T. McCarthy), 1984.

[3] HILL, J., et al., System architecture directions for networked sensors. In: Architectural Support for Programming Languages and Operating Systems, (2000) 93-104

[4] KAMEAS, A., et al., An Architecture that Treats Everyday Objects as Communicating Tangible Components. Proc. PerCom03, IEEE, Forth Worth.

[5] MAVROMMATI, I, KAMEAS, A. End user programming tools in ubiquitous computing applications. Proc. of the 10th International Conference on Human - Computer Interaction 2003 (HCI-International 2003).

[6] MAVROMMATI, I., MARKOPOULOS, P., CALEMIS, J., KAMEAS, A. Experiencing Extrovert Gadgets. Proc. HCI 2003, Vol. 2, Research Press International, 179-182.

[7] MAVROMMATI, I., MARKOPOULOS, P. , KAMEAS, A. Visibility and accessibility of a component-based approach for Ubiquitous Computing applications: the e-Gadgets case. Proc. of the 10th International Conference on Human - Computer Interaction, (HCI International), 2003

[8] NEWMAN, W., et al., Designing for serendipity: supporting end-user configuration of ubiquitous computing environments, In Proc. DIS'02, ACM Press (2002), 147-156.

[9] ROMÁN, M. and CAMPBELL, R.H., GAIA: Enabling Active Spaces, Proceedings of the 9th ACM SIGOPS European Workshop, Kolding, Denmark, Sept. 2000.

[10] ROMÁN, M., KON, F. and CAMPBELL, R.H, Reflective Middleware: From Your Desk to Your Hand. IEEE Distributed Systems Online Journal, Special Issue on Reflective Middleware, July 2001

[11] WEISER, M. Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM, 36(7), July 1993, pp 75-84