Requirements for an ubiquitous computing infrastructure

Sergio Maffioletti

Department of Informatique, Fribourg University

Email: Sergio.Maffioletti@unifr.ch

Сержио Маффиоллети – аспирант кафедры информатики университета в Фрибурге (Швейцария). Родился 11 сентября 1971 года в г.Бергамо (Италия). Закончил факультет информатики Миланского университета в 1998 году, дипломная работа посвящена интеллектуальным агентам. Участвовал в научно-исследовательском проекте Европейского Союза Larflast (Learning Foreign Language Scientific Terminology). Организатор студенческой лаборатории по параллельным процессам и искусственному интеллекту. Разработал курс «Язык программирования Java для Интернет-приложений» для учителей средних школ. Сфера научных интересов: архитектура программного обеспечения, информационные агенты, Интернет и другие гетерогенные информационные системы.

Вездесущие компьютерные технологии это образец взаимодействия между людьми и компьютерами. Их цель дать возможность пользоваться услугами компьютерных систем там, где этого пожелает пользователь. В статье представлены принципы разработки основы инфраструктуры для применения вездесущих компьютерных технологий. Причиной создания подобной инфраструктуры является формализация общей методологии разработки прикладных программ, основанных на представлении об интерактивной среде.

Всюдисущі комп'ютерні технології це зразок взаємодії між людьми і комп'ютерами. Їх мета дати можливість користуватися послугами комп'ютерних систем там, де цього побажає користувач. У статті представлені принципи розробки основи інфраструктури для застосування всюдисущих комп'ютерних технологій. Причиною створення подібної інфраструктури є формалізація загальної методології розробки прикладних програм, заснованих на уявленні про інтерактивну середу.

Ubiquitous computing is an emerging paradigm for interactions between people and computers. Its aim is to break away from the desktop computing to provide computational services to a user when and where required. In this paper we present the design criteria for an infrastructure platform for ubiquitous computing applications. The motivation for building such infrastructure is to formalize a common design methodology for developing application based on the notion of Interactive Environment.

1. Introduction

The way people used to think about computers and, as consequence, the way they interact with them, is always determined by the constrains the technology impose.

The more the technology availability and reliability can be taken for granted, the more the users change the nature of their requirements. In the early days of computation, where the technology support was rather rough and unreliable, users expectations were technology-driven. More performances and more reliable hardware was the customer's entire requirement. Nowadays technology is entering in its "mature phase" [5]. This means we can start taking technology for granted, and customers tend to change their expectations focusing on their everyday requirements, like the freedom to move around, the freedom to chose the medium used to attend the services offered by this technology.

The continuous changing of the relation between the technology and the users alter the role of technology in our lives.

In the past fifty years of computation there have been two great trends in this relationship: the mainframe and the PC relationship.

Today Internet is carrying us through an era of widespread distributed computing toward the relationship of ubiquitous computing [9] characterized by deeply imbedding computation in the world. The vision of ubiquitous computing, first expressed by Weiser [8], is to break away from the desktop computing to provide computational services to a user when and where required.

The ubiquitous computing era will have lots of computers sharing each of us. Some of these computers will be embedded in walls, chairs, clothing, light switches, cars. They will operate in the background of our life, in a transparent and not intrusive way. This vision will change not only the way we'll interface with computers but, also, the class of applications deployed for this scenario.

Actually applications deeply rely on computer's structure: desktop applications are conceived to be executed by a PC with a standard input system like keyboard or mouse and a rather simple output system like a monitor; distributed applications rely on a presence of more computers networked together. None of them have users as a central notion of their model.

Developing applications for UbiComp scenario means to deal with completely different constrains and functionalities determined by the user expectations, like the availability of services wherever, whenever and in any for the user require.

This is a rather important shift in HCI domain: applications integrated in our reality and realized by the interaction between one or more computing devices. We'll be able to use devices of different forms and functionality's; we'll interact with them in a more natural way (speech, vision, movement, facial expressions...).

The motivation for building these systems is to bring computation into the real, physical world to support what is traditionally considered non-computational activity; to allow computer to participate in activities that have never previously involved computation and to allow people to interact with computational system the way they interact with other people: via gesture, voice, movement, and context.

The main challenge for the success of these systems is the design of smart user interfaces and software that allows for ubiquitous and easy access to personal information and that is flexible enough to handle changes in user context and availability of resources; in one word an infrastructure. In this paper we present the basic requirements for an UbiComp infrastructure focused on defining new models for the development of interactive environment applications.

2. Ubiquitous computing system

Ubiquitous computing (UbiComp) denoted the universal availability of computation throughout multiple systems in the user's environment.

A ubiquitous computing system consists of a heterogeneous set of computing devices; a set of supported tasks; and some infrastructures the devices may rely on in order to carry out their tasks. Differently form desktop systems, the emphasis is on combining software components to provide services to the user. With an UbiComp system we are concerned not only with software services but also with devices and how to combine them.

Analyzing an UbiComp system we're interested in classify the main functionalities it supplies differently from the standard desktop systems. Weiser's [8] classification of an UbiComp system is based on two fundamental attributes:

- Ubiquity: interaction with the system is available wherever the user needs it.
- Transparency: the system is non-intrusive and is integrated into the everyday environment.

According with this classification Abowd [6] identified two dimensions that provide a rather clear boundary for UbiComp systems and express the relationship with other emerging research areas:

- User mobility: reflect the freedom the user has to move about when interacting with the system.

- Interface transparency: applies to the system's interface and reflects the conscious effort

- end attention the system requires of the user, either for operating it or for perceiving its output.

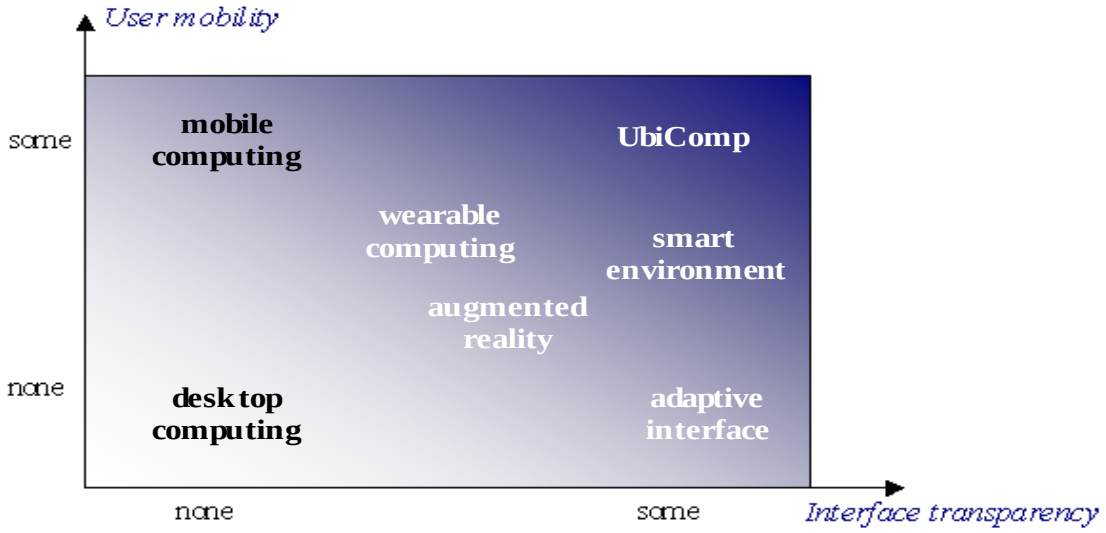

Figure 1 present an ontological framework based on the two dimensions defined above. This framework allows characterizing other research streams relevant to HCI and ubiquitous computing. As we may see UbiComp system tries to maximize the user mobility and the interface transparency. In contrast desktop computing offer no user mobility and, for the most part, no transparency to the end user.

3. Interactive environment

Research in ubiquitous computing is toward the development of application environment able to deal with the mobility of both users and computing devices. The vision of ubiquitous computing relies on the presence of environments enriched by computers embedded in everyday objects (blackboards, table, chairs...) and by sensors able to catch information form the context.

It is then an important requirement for UbiComp applications to provide a support environment in which specialized *computing instruments*[1] can be accommodated and integrated into existing application contexts.

Our interest is in Interactive Environments, defined as “Intelligent Environments”[3] or “Cooperatives Buildings”[7], conceived primary for workgroup interaction. These environments represent the spatial boundaries of applications integrated in our everyday context; they represent the physical space where the applications are placed and executed.

Applications developed for these environments have two dimensions: a service dimension that represents the number of services available in the system and a device dimension that represents the number of devices incorporated in the environment.

Interactive environments have specific computational properties that generally distinguish them from other computational systems. They have a large number of hardware and software components that need to cooperate; they tend to be highly dynamic and require reconfiguration and resource management on the fly as their components and inhabitants change and as they adjust their operation to suit the learned presence of their user.

Even though each interactive environment is created in its own way for its own purpose, they are generally built out of similar components.

Intelligent Environments defined in the Metaglue [3] project are an example. Intelligent Environments contain a multitude of subsystems comprising their perceptual interfaces, software applications, hardware device connections, and mechanisms for internal control.

[1] With ``computing instrument'' we refer to both devices and sensors. It identifies the abstract idea of computers integrated in our everyday environment

Figure 1: An interactive environment.

Figure 2 shows an example of an interactive environment. The application context represent the logical area where the application takes place and corresponds to one of the physical regions composing the interactive environment (these regions may vary in dimension from a single room to an entire building). A federation of computing instruments is a set of authenticated instruments cooperating with each other. The federation represents the group of instruments belonging each time to a specific application context.

The application will use the federation's topology in order to determine the service-device association. Hence an application is realized by the cooperation between services associated to different devices belonging to the same application context.

4. Common middleware

The few existing integrated multi-device computer environments today tend to be highly specialized and based on application-specific software.

Applications developed for interactive environments should be able to interconnect and manage large numbers of disparate hardware and software components. They should operate in real-time; dynamically add and remove components to a running system without interrupting its operation; control allocation of resources; and provide a means to capture persistent state information. Frequently these components are not designed to cooperate, so not only they must be connected, but also there is a need to express the "logic" of this interconnection. In other words, inter-component connections are not merely protocols, but also contain the explicit knowledge of how to use these protocols. Thus, viewing the connections simply as an application programming interface is insufficient. Cooperation among different applications is also difficult to achieve without a common platform. In order to model applications in this domain we need to define a common design methodology based on new paradigms independents from the technology. We are investigating a model to abstract the main components of an UbiComp system in order to formalize the development of interactive environment applications. These components may be classified into three abstraction layers:

- **Physical** deals with technological constrains.
- **Middleware** defines structure and the cooperation of abstract services.
- **Application** concerns the user interfaces.

Thank to these abstractions a middleware will present a uniform access abstraction for different ubiquitous devices, allowing them to interact and cooperate. This allows us to write applications scaling both on services offered and on devices composing the system. The model is not intended to be used for a single application, but to provide a standardized view of basic interactive environment functionality.

Functionalities

The main requirements that arise from the idea of a common middleware are:

- **Mobility:** First of all, the middleware has to deal with the mobility of the users and other physical objects. Therefore it must support mobile wireless communication and small mobile computing devices, such as PDAs and laptops.
- **Heterogeneity:** Like in the Nexus [1] project, a common middleware has to accommodate a large variety of application environments, ranging from areas covering whole nation to areas just covering an office. This leads to a very heterogeneous system, which has to support different network technologies and different tracking systems for example. Further aspects of heterogeneity come from the different services that rely on the presence of this middleware and the various input and output devices that have to be supported.

- **Scalability:** The middleware has to scale well for both a large number of cooperating services, which realize the application in each application context, and a large number of devices involved each time the application is used. Services represent the logical dimension of the application, while devices represent its physical dimension.

Existing projects like Oxygen [2], Nexus [1], Beach [7] and Metaglue [3] address the functionalities stated above as well but do not consider higher level service classification. Applications in these projects cannot rely on a suitable abstraction layer describing their functionalities.

What we need, instead, is to allow applications to define their own ontology for classifying resources they may offer and require. These high level concepts will be associated to low-level service implementation by the base infrastructure according with the constrains defined in each application context. This allows both the application and the instruments to use a high-level service description for interaction.

In such a way each computing instrument is also able to roam from one application context to another (even if the other is using another ontology) without changing its service description.

5. Conclusions

In this paper we have presented the general idea of a middleware that provides the basic functionality's for modeling interactive environment applications. This middleware will allow such applications to be created much more easily as they can rely on a common infrastructure. The platform maintains specific models for certain areas of the real world, which allow a user to access information, or services by their spatial position or through real world objects. A middleware for UbiComp applications aim at answering the need to provide a unified vision of the different computing functionality's issuing from convergence between information and communication technologies.

6. Reference

1. Fritz, Hohl and Uwe Kubach and Alexander Leonhardi and Kurt Rothermel. Next Century Challenges: Nexus – An Open Global Infrastructure for Spacial Aware Applications. ACM, 1999.
2. Laboratory for Computer Science and Artificial Intelligence. MIT project Oxygen. www.oxygen.les.mit.edu/, June 2000.
3. Michael H. Coen and Brenton A. Philips and Nimrod Warshawsky and Luke Weisman and Stephen Peters and Peter Finin. Meeting the Computational Needs for Intelligent Environment: The Metaglue System. Submitted to MANSE99, 1999.
4. Elizabeth D. Mynatt. Everyday Computing. GVU Center and College of Computing, Georgia Tech. March 1999.
5. Donald A. Norman. The Invisible Computer. The MIT Press, Cambridge, Massachusetts 02142, 1999.
6. D. Salber and A. K. Dey and G. D. Abowd. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Par adigm. GVU Center and College of Computing, Georgia Tech. 1999.
7. Norbert A. Streitz and Jorg Geissler and Torsten Holmenr. Roomware for cooperative buildings: Integrated Design of Architectural Spaces and Information Spaces. In N. A. Streitz, S. Konomi, H. J. Burkhardt, editor, LNCS 1370, Proceedings of the First International Workshop on Cooperative Buildings, pages 4-21, Darmstadt, February 1998.

8. M. Weiser. Some Computer Science Issue in Ubiquitous Computing. Communications on the ACM, 36(7): 75-84, July 1993.
9. M. Weiser and J. S. Brown. Design Calm Technology. PowerGrid Journal, July 1996.