An activity-centered ubiquitous computing framework for supporting occasional human activities in public places

Helder Pinto

Information Systems Department

School of Engineering, University of Minho

Supervisor: Rui J. José

Sep 12, 2008

Acknowledgements

This is the most wanted moment of a research work – reaching the end of a long and gratifying journey and thanking those who accompanied me in this work, those who alleviated pain in some occasions or shared some good moments that research can provide. I thus acknowledge:

Rui, for his engagement, professionalism, wisdom, and patience. Rui has, along these years, strongly influenced the way I reason, I write, and, above all, my critical thinking. Rui is great at putting in question what we believe or want to be unquestionable and I am deeply grateful for it.

Simona and Sílvia, for all their love, confidence, and understanding, and for all the joy they bring to my life.

My parents, Mário and Rosa, and my brother, Frederico, who have always been enthusiastic about my work.

Those who closely followed my work, for all their friendship and interest, specially Adriano Moreira, Ângelo Moreira, César Ariza, Filipe Meneses, Francisco Bernardo, Helena Rodrigues, Jason Pascoe, Noé Vilas Boas, and Paula Monteiro.

Those who have somehow contributed to the deployment of the several user studies, particularly Ângelo Moreira, César Ariza, Domingos Freitas, and Jorge Figueiredo, and all the subjects of the user studies, for their availability and interest.

The staff of the 2005 School of Engineering Week, Vila Flor Cultural Center, and Interacção 2006, for their availability and support to the user studies.

Ubisign, for the support given in these last months, and, finally, Fundação para a Ciência e a Tecnologia, for the financial support, without which all this work and results would be impossible.

Abstract

An activity-centered ubiquitous computing framework for supporting occasional human activities in public places

A major challenge to ubiquitous computing system designers is the provision of walk-up-and-use solutions for supporting activities performed by occasional visitors to a particular place. When arriving for the first time to a particular place, occasional visitors have little or no idea about what the local environment is providing to support their activity. Furthermore, this support has to be self-explainable and quickly learnable, as occasional visitors are not prepared to interact with an unknown system and do not have time to spend understanding and learning how to use new tools.

Ubiquitous computing environments promise to transparently support people in their daily activities by leveraging computing resources existent in the physical environment. Ubiquitous computing can greatly enhance the experience of occasional visitors to public spaces, by offering effective and transparent means for achieving their activities. Moreover, ubiquitous computing interaction artefacts are becoming increasingly cheap, thus allowing for widespread availability throughout public spaces. However, there is still much to do to achieve the vision of a computing system that requires little or no attention at all, so that humans can use the computer unconsciously. Ideally, people should perform an activity requiring computing tools as they perform any other activity, by focusing on the activity itself, and using the computing tool as naturally as other tools.

There is thus the need to center the design and development of ubiquitous systems in the human activity, in order to bring computing closer to people and to transparently support activities that take place in the physical world. This work thus follows an activity-centered approach to ubiquitous computing and contributes with Activi-

tySpot, an activity-centered conceptual and software framework targeted at providing ubiquitous computing support for occasional visitors to public spaces. The conceptual framework aims at modelling human activity and user interaction with the ubiquitous computing system. Undertaking an activity-centered approach to ubiquitous computing system design requires an understanding of how humans think about and carry out their activities. Therefore, this research is grounded on previous work on human activity analysis, namely Activity Theory, a conceptual framework for analyzing human activity developed during the twentieth century.

The software framework includes a ubiquitous computing infrastructure for providing the actual support to occasional visitors, tools for deploying ubiquitous computing solutions by non-computer-expert public space administrators, and a software library for developing the support to new activities.

Both the conceptual and software framework have been evaluated by a series of end-user studies which showed that ActivitySpot is effective for walk-up-and-use systems, making user interaction with a ubiquitous computing system almost as natural as interacting with other everyday tools. The majority of users clearly reported that ActivitySpot fostered learnability and usability. The choice of using elementary, everyday interaction means with a simple stimulus-response interaction model was also fundamental in the success with end-users. Moreover, the ActivitySpot software framework enables rapid development of the support for new activities and actions, by means of a software library for developers, as well as it eases the deployment and configuration of that support, by means of a graphical user interface authoring tool for public space managers.

Resumo

Plataforma de computação ubíqua, baseada em actividade, para suporte a actividades humanas ocasionais em espaços públicos

Um dos maiores desafios para quem desenha sistemas de computação ubíqua é o fornecimento de soluções que favoreçam a interacção espontânea, sem treino prévio, sobretudo as destinadas a suportar actividades realizadas por visitantes ocasionais de um determinado lugar. Quando chegam pela primeira vez a um determinado lugar, os visitantes sabem pouco ou nada sobre o que é disponibilizado para facilitar a sua visita. Além disso, o suporte fornecido tem de ter características que permitam uma aprendizagem rápida da sua utilização, visto que os visitantes ocasionais não estão preparados para interagir com um sistema desconhecido e não têm tempo para aprender a usar novas ferramentas.

A computação ubíqua promete ajudar, de modo transparente, as pessoas nas suas actividades diárias, tirando partido dos recursos de computação existentes no ambiente físico. A computação ubíqua pode deveras melhorar a experiência dos visitantes ocasionais de espaços públicos, oferecendo-lhes meios transparentes e eficazes para levarem a cabo as suas actividades. Além disso, os artefactos de computação ubíqua estão a tornar-se cada vez mais acessíveis, havendo pois condições para uma disponibilidade alargada na generalidade dos espaços públicos. No entanto, há ainda muito a fazer para concretizar a visão de um sistema de computação que requeira pouca ou nenhuma atenção por parte dos utilizadores. Idealmente, as pessoas deveriam executar uma actividade que requeira ferramentas de computação com a mesma facilidade com que executam outras actividades, focando-se na própria actividade e utilizando as ferramentas de computação tão naturalmente como utilizam outras ferramentas.

Há pois a necessidade de centrar o desenho e o desenvolvimento de sistemas ubíquos na actividade humana, de modo a aproximar a computação das pessoas e a suportar de maneira transparente as actividades que têm lugar no mundo físico. Este trabalho segue, portanto, uma abordagem centrada na actividade e contribui com o ActivitySpot, uma plataforma conceptual e de *software*, centrada na actividade, destinada a fornecer suporte de computação ubíqua para visitantes ocasionais de espaços públicos. A plataforma conceptual pretende modelar a actividade humana e a interacção com o sistema de computação ubíqua. A opção pela abordagem centrada na actividade requer o entendimento sobre como os humanos pensam e executam as suas actividades. É, pois, por isso que esta investigação é baseada em trabalho prévio na área de análise de actividades humanas, nomeadamente na Teoria da Actividade, um modelo conceptual de análise da actividade humana, desenvolvido durante o século XX.

A plataforma de *software* inclui uma infra-estrutura de computação ubíqua destinada a suportar as actividades de visitantes ocasionais, ferramentas para a instalação de soluções de computação ubíqua por parte de administradores de espaços públicos que não dominem necessariamente ferramentas computacionais, e uma biblioteca de *software* para o desenvolvimento do suporte a novas actividades.

Ambas as plataformas foram avaliadas por uma série de estudos com utilizadores. Estes estudos demonstraram que o ActivitySpot é eficaz em situações de utilização espontânea de sistemas que não tenham sido objecto de treino prévio, tornando a interacção com um sistema de computação ubíqua tão natural como a interacção com outras ferramentas do dia-a-dia. A maioria dos utilizadores relatou claramente que o ActivitySpot fomentou a fácil aprendizagem e utilização. A opção pela utilização de meios de interacção elementares e de uso diário e de um modelo de interacção simples, baseado no conceito de estímulo-resposta, foi igualmente fundamental no sucesso obtido com a utilização do sistema. Além disso, verificou-se que a plataforma ActivitySpot permite o rápido desenvolvimento de suporte a novas actividades e acções, através da biblioteca de *software*, assim como facilita a instalação e configuração do suporte às actividades, por meio de uma ferramenta gráfica de edição destinada a gestores de espaços públicos.

Table of contents

List of figures

List of tables

List of listings

Chapter 1

Introduction

Public spaces are the focal point of a vast range of human activities, such as entertainment, education, health or shopping, and a promising scenario for the deployment of ubiquitous computing systems. Whereas some of the activities that people may wish to perform at a public place are not particularly tied to any specific environment (e.g., reading e-mail at the museum cafeteria), there is a also a vast range of activities that can only be physically achieved or acquire special relevance in a specific place (e.g., visiting a relative at the hospital). These activities are characterized by a strong association with the specific social and physical setting in which they are meant to take place.

Some people are recurrent users of the places where activities occur (e.g., local workers) while others go there occasionally for very short-term work, for achieving some formality, for meeting with somebody, or just for entertainment. Particularly challenging, both from the point of view of actors – those who are involved in the activity – and facility managers – those who provide the means for supporting activity execution – are activities performed by *occasional visitors*, i.e., by people that are not used to live or work in a place and that occasionally pass by. When arriving for the first time to a particular public space, these visitors have little or no idea about the physical setting nor about the resource infrastructure that such an environment may provide to assist activities. These users need help to easily orient themselves in the physical environment, to identify the resources (humans or artefacts) available for achieving the activity, and to perceive how to interact with the available resources.

1.1 Motivation

Public spaces, in general, are designed and instrumented to provide some assistance to their visitors. They may have wall signs, panes, public digital kiosks, staff members, brochures, receptacles for comments and suggestions, etc. However, that type of support is normally targeted to the functional aspects of the space and very limited in providing people with a personalized and rich view of how the space can assist them with their needs and enhance the execution of the activities they intend to perform at that place. I next detail each of the main limitations I identify in the current support (either physical or digital) to occasional visitors to public spaces:

- **Lack of personalization**. People visiting a public space are not distinguished by their motives, context, preferences or impairments; everybody gets the same, mostly limited impression of the environment.
- **Weak interactivity**. Although most public spaces provide means for gathering visitors' opinions or wishes (surveys, complaints book, etc.), these often disrupt the visiting experience or are too formal or sanctioning. People usually prefer more spontaneous ways of interaction.
- **Weak integration**. Many public spaces do not provide an integrated view to its users. The several available types of resources are not conceptually or even visually coherent with each other. In other cases, the different means used for conveying the same information employ a different speech.
- **Lack of activity-centered character**. Assistance in public spaces is generally targeted at the functional aspects that are common to most of the activities that can be carried out there. Instead, it should be centered on the specific activities the space allows for, with specific support for each possible activity. For example, a museum visitor interested in visiting only paintings from a specific artist will hardly find support to easily get to those paintings. Instead, he or she will have to browse throughout the whole available paintings or visiting instructions. Furthermore, assistance usually does not consider non-standard users. For example, the available

assistance in most public spaces will be of little or no help to security inspectors or visitors with physical impairments. Assistance was not targeted to them nor to their type of activity.

- **External location-based assistance**. Information and communication technologies have brought innovations in personal appliances, such as mobile phones and personal digital assistants, able to assist, in a certain way, visitors to public spaces. For example, such devices, when coupled with location technology based on wireless cell information or GPS, can provide map orientation or access to mobile location-based services (e.g., finding a nearby restaurant). However, although potentially available anywhere the user goes and providing the user with a vast amount of location-based information, this assistance is poor regarding the specificities of a particular physical space. For example, mobile location-based services do not support a visitor willing to know the estimated waiting time in a queue at a restaurant. Their service is too generic to provide real value to satisfy specific, space-related user needs, because it is normally just based on a centralized information repository (in the device or somewhere in the Internet). Mobile location-based services are currently not able to dynamically adapt to nearby resources and enhance its capabilities in order to provide rich and value-added assistance to visitors [Pinto 05].

In summary, public spaces are still not providing its visitors with an integrated, rich, and personalized support for the activities they may carry out there. There is the need for solutions that successfully integrate the physical environment with its occasional users, placing the focus on activities.

1.2 Towards a framework for activities performed by occasional visitors to public places

This work is based on the assumption that an adequate assistance to visitors is more likely to be achieved if there is some type of local infrastructure capable of integrating on the one side local resources and knowledge about the physical environment and, on the other side, visitors, their resources, and knowledge about their activities. Many public places are currently instrumented with devices such as the already mentioned digital kiosks, public flat screens, radio-frequency readers, voice synthesis and recognition, and many other types of sensors and interaction devices. These devices, altogether, are becoming the realization of what Mark Weiser once called ubiquitous computing [Weiser 91]. Ubiquitous computing environments promise to transparently support people in their daily activities by leveraging computing resources existent in the physical environment. Ubiquitous computing can greatly enhance the experience of occasional visitors to public spaces, by offering effective and transparent means for achieving their activities, while providing a personalized support. Moreover, ubiquitous computing interaction artefacts are becoming increasingly cheap, thus allowing for widespread availability throughout public spaces. A public space of the future will be able to provide an integrated ubiquitous computing infrastructure, composed of many different types of interaction devices, sensors, and actuators targeted at the specific activities of visitors.

1.2.1 The need for a paradigm shift

During the last decades, humans have been using the computer as a new artifact or tool to achieve their activities. The computer relieves humans from doing tedious operations, which is particularly evident, for example, in industrial automation and control or in enterprise information systems. The personal computer has also been a successful tool for communication and entertainment. However, it has not yet achieved a comparably great success in supporting the mundane activities of everyday life. Unlike other human-invented tools, the personal computer is still too difficult to use [Norman 98], often disturbing users rather than helping them. This happens mainly because computing systems generally do not consider the characteristics of human activity. By imposing application- and document-centered models and unnatural interfaces, computers oblige humans to dedicate considerable efforts in manipulating the tool, rather than focusing on higher-level concerns. Paraphrasing Norman [Norman 98], it should be "*possible to have all the material needed for an activity ready at hand, available with little or no* *mental overhead: tools, documents, and information are gathered together into packages manually designed for the particular activities in which they participate*".

The computer tool requires so much attention and consciousness from the user that often an otherwise simple task becomes a laborious challenge. For example, setting up the computing environment and opening applications and documents required for some activity are not central to the activity itself and should be a simple, possibly automated operation. However, in many cases they require too much attention from the user. This characteristic may not have originated massive rejection in the adoption of personal desktop computers, but it may be decisive in the success of ubiquitous computing systems, particularly in public spaces. Visitors to public spaces will not sit in a comfortable chair in front of a desk, but will rather be moving within buildings or streets, possibly in a hurry, with one or both hands taken. Furthermore, ubiquitous computing systems are going to be used by every kind of people, and not only by a computer-educated population. Such a computing system cannot require too much attention, if possible any attention at all, so that humans can use the computer unconsciously [Satyanarayanan 01]. Ideally, people should perform an activity requiring computing tools as they perform any other activity, by focusing on the activity itself, and using the computing tool as naturally as other tool. There is thus the need to center the design and development of ubiquitous systems in the human activity, in order to bring computing closer to people and to transparently support activities that take place in the physical world [Abowd 00, Banavar 00, Christensen 02, Sousa 02, Norman 05].

Past ubiquitous computing prototypes aimed at assisting visitors to public spaces have been targeted to specific mobile devices such as PDAs or mobile phones. Besides limiting the user population to owners of a specific type of device, visitors have to previously download and install a fully fledged application into their device or to use a costly and cumbersome connection to a mobile internet site. Besides the low probability that such an application owns enough knowledge about the local resources available as the user roams throughout the space, it requires from users to concentrate their attention on a tiny application interface, possibly using both hands, and diverting them from the physical environment and the activity at hand. As stated by Abowd [Abowd 00],

> *The focus for the human at any one time is not a single interface to accomplish some task. Rather, the interaction is more free flowing, like our interaction with the rich physical world of people, places, and objects in our everyday lives.*

Ubiquitous computing is not about carrying a computer everywhere neither about keeping the focus on that computer; ubiquitous computing must avoid the focus on the computer [Weiser 91]. Therefore, it is likely that people will interact with ubiquitous computing environments by the means of many different, heterogeneous devices (spread over the physical space or carried by themselves), each device being used as a tool for accomplishing a particular aspect of their activity [Sousa 05]. This interaction, although distributed, is naturally perceived by people as part of the same activity, like we do, e.g., when we use different tools – knives, peelers, pans, etc. – for cooking.

1.2.2 Managing activity-centered, distributed, personalized interaction

Following the presented argument, the solution I propose for the occasional visitor assistance problem is based on these main assumptions: a) public spaces have a ubiquitous computing infrastructure providing most of the visitor assistance; b) visitors may interact with many different local devices during the same activity execution (distributed interaction); c) assistance to visitors is provided in an activity-centered manner. Notwithstanding, an all-embracing perspective of the problem must consider some additional aspects next described:

- The information and hardware resources a ubiquitous computing environment owns may not be sufficient for providing an effective assistance to visitors. This is particularly evident when personalization is desirable, because the local infrastructure is not likely to own information about visitors profile in advance. The local infrastructure may thus have to rely on the visitors' personal domain (e.g., profile information or personal interaction devices) for providing a personalized assistance.

- In a scenario in which every public space is equipped with a ubiquitous computing environment for assisting visitors, this kind of system becomes as commonplace as, for example, the electricity or heating infrastructures in a building. Nowadays, no one requires an electrician or a plumber in order to manage electricity or heating provisioning in the different rooms of a building. Control boards and switches can be easily manipulated by any adult person with minimum know-how. Likewise, I assume that ubiquitous computing infrastructures and visitor assistance are managed by individuals who are not necessarily ubiquitous computing experts, but mainly have a rather common computer know-how.

- As stated by the Greek philosopher Heraclitus, *the only thing that is constant is change*. In a society where everything is transient, public spaces follow every cultural and technological change. The nature and structure of activities in a public space change over time. Likewise, assistance to visitors' activities evolves, as new interaction devices become popular or new assistance features are required.

1.2.3 Thesis

The goal of this work is to develop an activity-centered ubiquitous computing framework targeted at assisting occasional visitors to public spaces. Specifically, this dissertation describes a conceptual framework for modelling activities and user interaction and its instantiation into a ubiquitous computing platform supporting the deployment and management of assistance to visitors. This approach gives to the activities supported by public spaces a central role in system conception and perception. Visitors, public space managers, and ubiquitous computing developers all share a similar perspective of how ubiquitous computing tools may help carrying out activities. This dissertation shows that:

> *Ubiquitous computing environments employing an activity-centered framework for user interaction and system design, based on a simple conceptual model and on the usage of elementary, everyday interaction devices, are able to: a) provide effective walk-up-and-use assistance to activities performed by*

occasional visitors; b) ease the task of public space managers in deploying or reconfiguring the support to activities, with the help of appropriate tools; and c) ease the task of ubiquitous computing specialists in developing the support for new assistance features.

1.3 Challenges

Demonstrating the thesis above involves dealing with many challenges, more or less central to the thesis statement. This section describes all these challenges, with particular focus to those which are more deeply related to the thesis statement. The next section discusses the research goals, indicating how challenges are addressed – some of these challenges are partially addressed, either because they are not central to the thesis or because they imply an effort which is beyond the scope of a PhD thesis.

1.3.1 Activity model

Past research on activity models for supporting the analysis, design, and development of computer systems (e.g., [Jacob 86, Myers 00]) shows evidence of its suitability for desktop scenarios. However, ubiquitous computing may be taking protagonism away from the desktop, changing the way systems are designed and built and, therefore, how activities are modelled. We understand activity model as the way a particular activity structure is represented in a human- and machine-understandable vocabulary, so that it can be both effectively communicated to users and implemented in a ubiquitous computing infrastructure. Representing how humans perform an activity is a difficult task, as people may have different mental models of the same activity. Furthermore, the informal and rather unpredictable nature of the activities addressed in this work severally affects the efficacy of more structured approaches to formalizing the steps that compose an activity, such as those used in workflow systems [Suchman 87, Bardram 97, Abowd 00]. There is therefore a trade-off between the need for a generic model of activity that can be instantiated by an activity-centered infrastructure for different application scenarios and the risk of imposing our view of activity on heterogeneous mind-sets.

Besides the psychological aspect, other factors increase the difficulty of modelling human activity. Contextual and personal factors are especially challenging. For example, disabled people have to perform an activity possibly in a very different manner; a person that arrives at the hospital reception may go there for different reasons – for visiting a relative, for equipment maintenance, etc. Furthermore, the resources that may be used for accomplishing an activity are everything but static: personal resources brought by visitors are expected to be transient and available only for the time the visit lasts; local resources may vary from one moment to the other, e.g., due to breakdowns. Collaborative scenarios add to the complexity of the activity model: role management, coordination of users' actions, and all the aforementioned issues characteristic of individual activities that acquire more complexity in collaborative settings.

Finally, as activities are seldom carried out without interruption, the activity model must cater for activity resumption with no mental overhead for the user. This adds state to activity modelling, but adds as well human-computer interaction concerns, as state must be expressive enough for both system activity recovering and helping users remember in what point the activity was left [Abowd 00].

In summary, we need to define an activity model that: a) is generic enough to be instantiated in any activity while not imposing a vision of activity that is so restrictive that it does not fit heterogeneous mind-sets; b) considers contextual and personal characteristics of activity, required resources, collaborative scenarios, and interruption/resumption; c) is simple enough to be used by public space administrators that have shallow computer knowledge and to be quickly learnt by ubiquitous computing developers, while being powerful enough to support a wide range of solutions.

1.3.2 User interaction model

This work assumes that the user population is composed of occasional visitors to a public space and that the interaction with the ubiquitous computing infrastructure is going to be distributed among several devices. We are thus dealing with people that have a sporadic or even a single contact with the ubiquitous computing infrastructure, and, therefore, have a very short time to learn how to use the available ubiquitous computing resources. Among the issues posed by this situation, there is primarily the initial interaction with newly arrived visitors, i.e., making them aware of the available support to activities. For example, as people see arrows in the streets or buildings indicating destinations, the local infrastructure should indicate to users the available support to activities and how to use it. This is mainly a non-system issue, but with significant weight in the system usability. Moreover, while users accomplish an activity, the system should provide them with a non-obtrusive support, i.e., avoiding interaction mechanisms that distract users from their activity. Preferably, the interaction model should not require from visitors a cognitive effort that goes beyond the reasonable for such a walk-up-and-use system. The reasonability of this effort is related with the required learning time and with how naturally system usage integrates with the activity. When designing for an activity, overall simplicity in features play a prominent role [Abowd 00]. Ideally, usage instructions should not be needed at all, if user interfaces are simple enough and self-explainable.

Heterogeneity is a ubiquitous characteristic of ubiquitous computing, particularly regarding user interaction. Ubiquitous computing environments may be instrumented with a broad variety of interaction devices and they take in people bringing their own heterogeneous personal devices. Using a device-dependent approach for designing user interaction is overwhelming, as new devices and interaction possibilities are constantly coming up. Powerful abstractions are needed for dealing with this heterogeneity. Furthermore, the user interaction model must help people feel that all interactions, whatever device is used, are integrated and all part of the same activity.

In summary, the user interaction model must: a) facilitate visible, non-obtrusive, and simple interaction means; b) provide abstractions powerful enough to deal with heterogeneity while not compromising simplicity of interaction; and c) make multiple interactions be perceived as being integrated and part of the same activity.

1.3.3 User interface management

After a ubiquitous computing infrastructure is deployed in a public space, additional challenges have to be considered: support to new activities may be needed, new interaction devices may become available, or simply current support may have to be reconfigured. This requires someone to manipulate a tool representing activities and devices in order to configure the ubiquitous computing user interface that is available to visitors, by adding new activities, new devices, associating devices to activities, etc. This is, to a certain extent, similar to graphical user interface management tools, in which graphical user interface widgets (e.g., buttons, list boxes, menus, etc.) are manipulated and associated to events and actions in order to build a user interface.

Desktop user interface management in general is usually assigned to computer specialists, because it requires specialized computer knowledge. Desktop user interface management is mainly done at software houses, where computer specialists abound. However, ubiquitous computing user interfaces require deeper specialized knowledge about the place and the activities and may have to be managed *in situ*, at the location where they are deployed. The problem is that for many public spaces, it may not be possible to hire specialists for managing a ubiquitous computing user interface. If we want people with common computer knowledge to manage a ubiquitous computing user interface, they have to be provided with very high level tools that ease their task. The main challenge here is what kind of tool, abstractions, and metaphors to use, both for representing and configuring interaction devices and the support for activities.

The usual desktop user interface management tools are not suited to ubiquitous computing, due to the distributed character of interaction and, above all, to the type of user interface, which is no longer the screen-keyboard-mouse solution, but is rather an amalgam of different types of devices, with different sizes (tending to smaller ones) and form factors. I am not referring only to mobile phones, but also to other modalities of interaction, e.g., voice recognition and synthesis, public displays, radio-frequency identification, gesture recognition, etc. We have thus to devise new user interface management tools dealing with new forms of interaction. These tools have to deal also with a new dimension in user interface management: the physical space. Interaction devices are spread all over the space and must be contextualized into that same space by the tool. Furthermore, such a management tool must keep the user away from the details of how interaction devices interoperate with the infrastructure. It is also expected that such a tool is extensible, i.e., able to support new interaction devices and other local resources without having to rebuild it.

Another challenge of such a tool is how to create the conditions for avoiding configuration or manipulation errors made by user interface managers, i.e., providing a tool with a user interface that is designed in a way that errors are impossible or at least difficult to be made. This is what Myers calls *path of least resistance*: leading users towards doing the right things and away from doing the wrong things [Myers 00]. Achieving a highly usable and low error prone tool and, at the same time, powerful enough to enable the deployment and configuration of useful support to activities is a major challenge.

In summary, the challenges posed by ubiquitous computing user interface management include: a) the identification of adequate tools, abstractions, and metaphors for non-specialists to manipulate infrastructure support to both simple and complex activities; b) dealing with new forms of interaction, while keeping the user away from the low-level details of interaction devices; and c) enabling easy evolution by the seamless addition of new interaction means and other resources.

1.3.4 Facilitating system development

As seen in the previous sub-section, a ubiquitous computing system supporting a particular public space is going to be frequently updated either with new functionalities or with new interaction devices. Besides the user interface management aspect, there is also the logic implementation problem, which requires the development of software specific to the activities being supported. For example, if a message posting functionality is required in a public space, software has to be developed to process incoming messages (e.g., through SMS), to store it into a database, and to allow later visualization through some output device (e.g., a public display). If later a microphone interface is added, new software has to be written for supporting voice message input.

Software development tasks should require a minimum effort in order to minimize system development costs and to facilitate the introduction of new functionalities and interaction devices. If developing a new functionality is intimidating or expensive, the support provided by ubiquitous computing spaces becomes sooner or later unsuited to visitor needs or obsolete. However, minimizing ubiquitous computing development costs requires a combination of characteristics that, *per se*, is not easily accomplishable. Firstly, developers must deal with a simple ubiquitous computing architecture that does not pose learning and development obstacles. All the different interaction devices should be effectively represented in an abstract manner, so that developers do not have to bother with low level operational details. Furthermore, developers should not care about how the infrastructure processes device manipulation and actuation, but rather focus solely on how their particular piece of functionality reacts to user interaction. Besides this clear demarcation of concerns, the infrastructure must provide tools that help reducing the amount of code that programmers need to produce when creating the support for a new functionality and thus speeding up development pace.

A nowadays indispensable characteristic of any software infrastructure is the degree to which it is extensible. Developers should easily extend the ubiquitous computing infrastructure here considered with new interaction devices or scenario-specific functionalities without needing to rebuild the infrastructure code. The challenge here is to define contracts between the infrastructure extension points and extension developers so that such contracts are easily learnable by developers and enable the development of extensions with different levels of complexity.

In summary, a developable ubiquitous computing infrastructure implies several challenges: a) the combination of developer tools characterized by an abstraction level enabling easy and rapid development of new ubiquitous computing functionalities with the need of making the most of the available interaction devices; and b) providing extensibility mechanisms that are easily learnable and that enable the development of extensions with different levels of complexity.

1.3.5 Integrating local infrastructure and personal resources

Providing a personalized experience to occasional visitors is a specially challenging issue to ubiquitous computing systems. An activity cannot be supported in a personalized manner solely by the ubiquitous computing infrastructure or by the personal resources. Local infrastructure and personal resources have to integrate with each other in order to achieve both a personalized support and a thorough exploration of the local resources. However, since users may have not previously visited the space, we cannot assume the existence of a local personal profile or information about a particular user and her/his resources. This makes integration with personal resources a very hard challenge, with implications at every step of the support to an activity. Firstly, visitors must be identified by the local infrastructure. This must be achieved for every interaction, whether the visitor carries a smart phone, wears a smart tag, or brings nothing at all. Furthermore, visitors have to consent in being identified, even if this identification is done, e.g., through a pseudonym. Assuming that visitors may employ different devices to interact with the infrastructure, there is the additional challenge of associating to a single identity all the different devices or interactions made through it.

The second integration challenge arises when the support to the activity requires the access to the users' personal resources (profile, personal equipment, context, etc.). The issues associated to this challenge can be grouped into resource access and resource management. Regarding resource access, there is first the problem of making personal resources known to the local infrastructure, i.e., mechanisms for personal resource publication when a visitor arrives to a ubiquitous computing environment. After this step, comes the problem of providing access to personal resources only by authorized parties. For visitors to trust a local infrastructure, there must be some form of physical or digital certificate as well as the guarantee that visitor privacy is respected. The access to the personal resources itself is possibly the major challenge, given the potential heterogeneity of profile and context services or personal appliances. The system must be able to deal with this heterogeneity in order to avoid putting aside visitors without the required resources.

The resource management issue is particularly pertinent in the case the support to an activity depends both on local and personal resources. Some activities may include tasks that are supported alternatively between the local and the personal domains (e.g., switching between a user device display for viewing personal, sensitive information and a wall-sized display for achieving better visualization quality for non-sensitive information). These transitions should occur smoothly, without disturbing the user, and, when user interaction is involved, with some cues driving the user's attention to the new interaction device.

In summary, integrating local and personal resources poses two main challenges: a) identifying user interactions and associating them to a single user identity; and b) accessing and making use of personal resources while dealing with heterogeneity and privacy.

1.3.6 Activity experience capture, sharing, and customization

The ways of performing an activity, or praxis, evolve over time. When getting trained at some activity, we share its praxis and at the same time this praxis is continued and possibly changed [Bødker 91a]. This feedback loop in the course of an activity forms the basis for a learning process embedded in the activity itself [Bardram 97]. The practitioners of one activity remember their own experience, share it with others orally or through documents, and introduce the modifications that best suit their needs. In the context of activities performed at a particular place, remembering previous experience or sharing it with others is especially helpful to visitors that do not return often to perform that activity or that are unexperienced at all. However, it is not trivial to describe experience in a human understandable manner and, especially, to choose – pre-hoc or ad-hoc – which events are sufficiently relevant to be captured. A solution may be to allow users to insert annotations into the experience record, so that they enrich the information that others (or themselves) may easily access later. However, the effectiveness of this solution may be severely affected by the additional effort it requires from users.

As people have their own way of doing things, and as part of a process of activity evolution, visitors may want to customize recurrent activities in a specific place, so that next time they perform the activity they are better supported or they transmit to others better ways of performing that activity. Activity customization involves several challenges: to specify which parts of the activity can be customized; to what limit this customization can be done; how to formalize the customization; and how this customization influences following activity instantiations.

In summary, the challenges that activity experience capture, sharing, and customization may be reduced to the following: a) how to capture and formalize previous experience description and, at the same time, making it human-understandable; b) how to enable and define the boundaries of activity customization.

1.4 Research overview

Among the challenges identified in the previous section, some are less central than others to the problem of supporting activities performed at a particular place. Moreover, those challenges are mostly by themselves a topic for a research project on its own. I have thus decided to focus my research on a restricted set of challenges, which I consider to be essential in achieving my vision of activity-centered ubiquitous computing in public spaces. The results of my research are the development of a conceptual model of activity and user interaction and a ubiquitous computing infrastructure based on it for supporting activities performed by occasional visitors. This framework is not only targeted at end users – the visitors – but as well at public space facility managers and ubiquitous computing developers.

Undertaking an activity-centered approach to ubiquitous computing system design requires an understanding of how humans think about and carry out their activities. I believe that the best approach to overcome the activity and user interaction modelling challenges is to ground my research on previous work on human activity analysis. A theoretical framework of human activity provides ubiquitous computing researchers with an agreed set of terms to describe activity and with concepts that drive them in the construction of systems that intend to support activity [Constantine 06]. Among several frameworks produced mainly by the fields of psychology and philosophy, I chose Activity Theory [Luria 76, Leontiev 78, Vygotsky 78, Leontiev 81, Wertsch 81, Engeström 87, Engeström 99b], a conceptual framework for analyzing human activity developed during the twentieth century, as the background for this work, based on its maturity acquired along several decades of research and its set of simple and solid concepts. These concepts, particularly those related to the structure of human activity (the levels of activity, actions, and operations), form the basis of the activity model I propose.

For modelling user interaction, I also followed an approach based on simple concepts. Given that user interaction with a ubiquitous computing system is done through multiple, heterogeneous means and, in many cases, with little common characteristics, I reduced user interaction analysis to basic human-computer interaction concepts: stimulus and response. I assume that, for a given stimulus through a given interaction medium, a response is produced, synchronously or not, through the same medium or through other medium or set of media. Interaction media are assumed to be elementary ones, such as voice input/output, gesture recognition, RFID tag reading, SMS, public screen display, etc.

The solution I propose for the ubiquitous computing user interface management challenge is based on the activity-centered conceptual model introduced above and on the usage of a graphical user interface authoring tool. Such a tool employs an activity-centered approach, so that the person in charge for the public space facilities can manage the local environment with activities in mind. The user – the facilities manager – thinks of activities, actions composing it, and interaction devices used for executing actions.

For example, when a new activity is going to be supported, the authoring tool has to provide the manager with all the necessary information about local resources (devices, sensors, available actions, etc.) so that she can decide how to leverage the resources in order to support visitors. Furthermore, such a tool provides graphical information about the physical structure of the public space (e.g., in the form of plans) and how the devices are spread all over the space, allowing the user to reorganize it if needed.

In order to ease the task of developing support for new actions, this work intends to provide software developers with high-level programming abstractions for representing stimuli and responses, with minimum dependence on the interaction medium, and following fundamental principles of framework design [Cwalina 05], such as offering a low barrier to entry (to ease the task of those willing to learn by experiment) or providing self-documenting object models (derived from the interaction and activity models described above). This goal is achieved by the means of a software library, described in Chapter 4.

The following subsections describe the method for validating the thesis and present the contributions of this dissertation.

1.4.1 Thesis validation

The evaluation of this research implies the validation of the thesis stated in section 1.2.3. This involves demonstrating that the proposed model of activity and user interaction: a) is implementable in a ubiquitous computing infrastructure; b) enables a visitor experience enhanced by the capabilities of ubiquitous computing, making the visitor feel effectively assisted in the activity in hands, without distracting him or her; c) is suitable for the needs of public space managers, i.e., it provides clear benefits regarding productivity and manageability in deploying or reconfiguring the support to activities, with the help of appropriate tools; and d) eases the task of ubiquitous computing specialists in developing the support for new assistance features.

The first validation goal is achieved by the implementation and usage of a ubiquitous computing infrastructure derived from the proposed interaction and activity models – the ActivitySpot infrastructure.

The evaluation of the end-user experience with ActivitySpot-based support to activities is accomplished by several user studies, based on real scenarios, which analyze: the compatibility of the proposed model with the user's conceptual model; how successful is user interaction (e.g., without the need for previous training or without distracting from the actual activity); and the degree of usefulness assigned by end-users to the proposed approach compared to other approaches (e.g., mobile phone application or public kiosk).

The public space management tool is evaluated by user studies with individuals who usually perform public space facilities management. These users are given a set of management tasks (e.g., configuring the support for a new activity or reconfiguring the support for an existing activity) which they have to accomplish. Their performance in achieving the tasks and the opinion regarding the tool is evaluated.

Finally, the evaluation of the software library for ubiquitous computing developers is done by analyzing the solutions developed for the end-user experiences, taking into account productivity and provided power, i.e., how much does the software library allow for.

Chapter 5 addresses in detail the thesis validation.

1.4.2 Contributions

This work targets the core of the challenges posed by the support to activities performed by occasional visitors. This work can contribute importantly to the area of ubiquitous computing by proposing a framework that facilitates the development and deployment of effective ubiquitous computing solutions. The main contributions of this work are:

- An *activity-centered conceptual framework* for ubiquitous computing environments. The framework is composed of an activity model and a user interaction model. The activity model, informed by Activity Theory and developed with simplicity of concepts as requirement, is intended to effectively support users of a ubiquitous computing system in public space scenarios, particularly when those users are occasional visitors. The interaction model is based on elementary, everyday interaction means, which fosters short learning time and usability. (see sub-section 1.3.2). Furthermore, the activity-centered character of the proposed framework makes user interaction in a ubiquitous computing environment as natural as interacting with other everyday tools and, as described in the next items, the model is successfully implemented and is effective in most of the issues around design and deployment of ubiquitous computing solutions for public spaces.

- A *run-time infrastructure* implementing the conceptual framework. This infrastructure is composed of a set of components and architectural abstractions over which the support to activities depends. The infrastructure also includes a run-time environment coordinating user interaction and managing the execution of activities.

- An *authoring tool for public space managers* for configuring the support to activities. This graphical user interface tool is based on the proposed conceptual model, representing activities, actions, interaction devices, and physical space. Its users can assign actions to activities, devices to actions, and represent device distribution over the physical space. Further configuration is also possible, such as defining context-based execution conditions for activities and actions. With this tool, public space managers with common computer knowledge can easily configure ubiquitous computing support to activities without having to care about the internals of the infrastructure (see sub-section 1.3.3).

- A *software library* for supporting the development of new actions. Developers do not have to care about how the activity is composed or where the interaction devices are disposed or even about the low-level interaction mechanisms (see sub-section 1.3.4). Developers only have to care about writing the code for stimulus reaction behavior, using high-level abstractions for representing stimuli and responses, with a weak dependence on the type of interaction medium.

1.5 Plan of dissertation

The remainder of this dissertation is organized as follows. Chapter 2 presents other works which are more or less closely related to mine, namely works on ubiquitous computing assistance to visitors, activity-centered computing, ubiquitous computing infrastructures, applications of Activity Theory, among others.

Chapter 3 details my proposed model for human activity and user interaction in ubiquitous computing environments. It begins with a deeper description of Activity Theory, relating it with the specificities of activities performed by occasional visitors, and showing how it informs the construction of the activity model I propose. This chapter completes with a description of the whole proposed conceptual framework, including the user interaction model.

Chapter 4 is dedicated to the realization of the proposed conceptual framework. This is done with three elements: a run-time infrastructure, a software library, and a graphical user interface authoring tool. The run-time infrastructure supports the deployment and execution of the actual ubiquitous computing support to activities. The software library provides the means for developing the support to new activities.

The authoring tool is targeted at public space managers for creating configurations of the support to new activities or to reconfigure existing ones. For each software framework element, I enumerate technical requirements, describe its implementation, and discuss how the implementation meets the requirements.

The method and the steps employed for validating my thesis are described in Chapter 5. Validation comprises three perspectives: the end-user perspective (actual visitors interacting with the system in a public space); the manager perspective (authoring tool users); and the developer perspective (software library users).

Finally, the dissertation concludes with final remarks and with possible opportunities for future work.

1.6 Summary

Ubiquitous computing is definitely a promising solution for supporting occasional visitors to public spaces, providing them with assistance in their activities. This thesis is based on the statement, shared by many authors, that the best approach to make ubiquitous computing truly transparent and natural to users is an activity-centered one. For this, I ground my work on an activity analysis framework developed during the twentieth century: Activity Theory. Activity Theory is still evolving and has been applied in different activity-centered approaches in several computer science domains. Besides the activity-centered character, I consider that ubiquitous computing solutions should be grounded on distributed user interaction with everyday, elementary interaction means (e.g., public screens, radio-frequency or magnetic cards, buttons, SMS, etc.), rather than on scenario-specific devices (the device-dependent fully-fledged application is the common example). Distributed interaction with simple devices is more similar to our everyday interaction with physical artefacts and more appropriate to the needs of occasional visitors to public spaces with no time for learning how to use a complex system. An activity-centered approach also enhances or, at least, alleviates the task of people who arrive to a public space and want support for the specific activity they have in mind, rather than having to deal with a functional perspective that is presented to every person, no matter what activity they are going to perform.

This vision poses several technical and conceptual challenges, with particular focus on: a) the activity and user interaction models, i.e., how to abstractly represent human activity and user interaction so that it can be effectively instantiated in any specific activity scenario; b) what solutions to provide when considering that a ubiquitous computing infrastructure in a public space must be manageable by staff people without ubiquitous computing expertise; c) how to design a ubiquitous computing framework so that it can evolve as the need for new solutions emerges, namely by allowing developers to easily extends the infrastructure with the support for new actions.

This work intends to produce an activity-centered conceptual framework for designing and deploying ubiquitous computing solutions for activities in public spaces. This framework is used to develop a software infrastructure for running the support to human activities and a set of tools for public space managers configuring that support and for ubiquitous computing specialists for developing the support to new actions. All these contributions are evaluated by user studies carried out for the three different framework user perspectives: the end-user perspective (actual visitors interacting with the system in a public space); the manager perspective (users of the authoring tool); and the developer perspective (users of the software library).

Chapter 2

Related work

My work deals with research challenges that cover several infrastructure and user interaction aspects, acquiring a multi-disciplinary character with connections to a broad set of the relevant research that has been done in the ubiquitous computing field.

This chapter is divided into eight main sections. The first one is dedicated to ubiquitous computing systems aimed at supporting visitors to public spaces. Next, I dedicate three sections to activity-based research, namely activity-based computing, activity modelling, and applications of Activity Theory. I then explore connections with the user interaction fields, with sections dedicated to distributed user interaction and end-user programming in ubiquitous computing environments. I finally describe relevant generic ubiquitous computing infrastructures and close the chapter with work on the integration between the personal domain and local infrastructures.

2.1 Assistance to visitors

Several systems have been proposed and made publicly available for assisting visitors in many different scenarios. Audio guidebooks for museum visitors are a pervasive example, available mainly as an audio substitute for conventional guidebooks. Later, multimedia guides were introduced and brought an additional sensory experience to visitors. Multimedia guides still suffer from the shortages of audio counterparts – the lack of context-awareness and personalization – and introduce the problem of excessive attention required from users manipulating the mobile multimedia device. Digital kiosks are in principle more usable than an application in a mobile multimedia device. However, due to its dimensions, they cannot follow visitors wherever they go and have to be shared among users.

I present here a set of ubiquitous computing systems which I consider to be relevant forward steps in enhancing visiting experiences. I divide them into three main groups: context-aware mobile applications; universal interaction systems; and non-conventional interaction models.

2.1.1 Context-aware mobile applications

By context-aware mobile applications I mean context-aware, rich client applications running on a mobile device such as a PDA or a smart-phone. All the following systems use some context variable (mainly location) to generate the information presented to visitors.

Cyberguide

The initial prototypes of Cyberguide [Abowd 97] were designed to assist a very specific kind of tourist – a visitor in a tour of the GVU Center Lab at the Georgia Institute of Technology, during their monthly open houses. The prototype application was designed to run on commercially available PDAs and pen-based PCs, in which context-awareness simply means the current physical position and orientation of the Cyberguide unit. Cyberguide users interact with a rich application, composed of menus, buttons, and information panels dependent on the user location.

Cyberguide is divided into several independent components, personified in terms of the people a tourist would like to have available while exploring unfamiliar territory: cartographer (maps of the physical environments that the tourist is visiting); librarian (realized as a structured repository of information – local to the device – relating to objects and people of interest in the physical world); navigator (realized by a positioning module that delivers accurate information on tourist location and orientation); messenger (realized as a set of wireless communications services).

Outdoor Cyberguide prototypes were also developed, assisting a tour of the Georgia Tech campus and surrounding neighborhoods or assisting tourists in pursuit of refreshment at neighborhood establishments in Atlanta.

The Conference Assistant

The Conference Assistant [Dey 99] is a context-aware application for assisting conference attendees and presenters, namely by helping them to decide which activities to attend, to provide awareness of the activities of colleagues, to enhance interactions between users and the environment, to assist users in taking notes on presentations and to aid in the retrieval of conference information after the conference concludes.

The Conference Assistant application runs on platforms such as laptops and handheld devices. It uses several dimensions of context and personal data, namely identity, preferences, time, location, and activity. For example, on the conference schedule, certain papers and demonstrations are highlighted to indicate that they may be of particular interest to the user. When the user enters the conference room, the Conference Assistant automatically displays the name of the presenter and the title of the presentation, allowing the user to create notes of her own to attach to the current slide or Web page or to control the presenter's display at the end of the presentation, bringing up a question and allowing everyone in the room to view the slide in question.

GUIDE

GUIDE [Cheverst 00] is a rich, context-aware application specifically designed to assist visitors to the city of Lancaster. The information presented to visitors is tailored based on the visitor's user profile and contextual information, including the physical location of the hand-held unit where GUIDE is running.

The user interface to GUIDE is based around a modified browser metaphor. GUIDE users can access location-aware information, access interactive services (e.g., booking of hotel accommodation), send and receive textual messages, and create a tailored tour of the city, which dynamically adapts to visit durations and schedules.

The network infrastructure that is used by the GUIDE system comprises a number of interconnected wireless cells. A single cell-server is associated with each cell and this server is responsible for broadcasting information to GUIDE units as they enter the cell-server's zone of coverage.

Hermes

Hermes [Driver 04] is a software framework for mobile, context-aware trails-based applications targeted at visitors to public places. A trail can be thought of as a collection of locations, together with associated information and activities, and a dynamically reconfigurable recommended visiting order. The trail is a collection of connected locations rather than a strict sequence since it may contain alternative sub-routes to cater for such variables as different modes of transport or other user preferences. Trail activities can be either mandatory or optional, and each activity has a priority value which is used to rate its importance relative to other activities.

Hermes was evaluated as an application for student assistance in performing campus-based activities in their first day at the college. Students used their Hermes application to create a personalized route from the student registration point, to any library on campus, and on to a specific lecture theatre. Each point on the trail has an associated task (and subtasks) i.e., register, attend introductory lecture with course director. The trail is presented on an augmented digital map of the campus with routes drawn between each activity point, and the user's current location denoted. At all times, the users personal and environmental context can dictate that the trail being followed is dynamically reconfigured to reflect the relationship between their trail activities and the current state of their environment.

2.1.2 Universal interaction

The purpose of universal interaction is to provide mobile users with a single interface – a kind of universal controller – to all the ubiquitous computing services and devices available in the physical environment, assuming that all these resources implement the same type of interface. Visitors to public spaces using a universal interaction tool can explore the ubiquitous computing environment for information and services needed for their activity.

Universal Interaction System

The Universal Interaction System [Hodes 98] uses location-based services as the basis for allowing a client device to function as a "universal interactor", i.e., as a device that can discover the control interface of any other devices in the environment and adapt itself to control those devices. Universal interaction is addressed by avoiding assumptions about the interface with devices, by allowing services to transfer an entire graphical user-interface description to the client, and also through the use of a transduction protocol that maps the control functions provided by the service into a user-interface supported by the client device. The service interface specification is an XML document describing the user-interface elements and how they map to service requests and responses. The client interprets this specification and generates platform-specific user interface widgets.

Universal Information Appliance

The goal of the Universal Information Appliance (UIA) [Eustice 99] effort is to create an environment in which a single device can serve as a person's portal into the digital and electronic domain. The user's interface is presented by a wearable computer, which the user presumably carries whenever he or she desires to be part of the electronic network. In place of a platform-specific user-interface rendering, the UIA effort focuses on building virtual machinery capable of rendering user interfaces on any device by translating a common expression for user interfaces into a platform-specific implementation.

This approach is similar to the work of Hodes et al., though it includes additional functionalities, such as providing local database access, storing the user's profile information, maintaining the soft state for the user's current applications, and holding a cache of knowledge to which the user wants immediate access. The UIA interface specification language – MoDAL – also differs in that it allows for flow control structures and local variable assignment as well as allowing mobile clients to retrieve updates to applications without having to retrieve a full copy of the updated application.

Cooltown

The Cooltown project [Kindberg 00] adapts the web infrastructure to support nomadic users, providing location-specific services in the places that such users visit and providing interaction with the things that they encounter. Cooltown can be seen as a universal interaction infrastructure, because it is based on a standard web browser, thus allowing for universal access to web resources. Cooltown applies the notion of web presence – being accessible over the web infrastructure – to people, places, and things. Things become web-present by embedding web servers in them or by hosting their web-presence within a web server. Places become web present by organizing web things into collections under the management of a web service – a place manager. People become web present by offering home pages with services to facilitate communications between individuals and by offering information via location-specific place managers.

The typical user experience Cooltown seeks with web presence is that of collecting links to points of web presence as users encounter them in the physical world. The usual Cooltown scenario is a museum visit. As visitors tour the museum, their PDA receives web URLs from wireless beacons – small infrared transceivers located close to pictures or sculptures. The URLs link into a web of information about the items. Using the PDA's web browser, visitors can read or hear about the artist or the work and about related art works in the museum. The URLs can also be used to select reproductions of the artwork from the museum's online store. The museum staff uses the same URLs for inventory control as the URLs point to the object's point of web presence.

2.1.3 Non-conventional interaction models

As seen in the previous sections, ubiquitous computing prototypes aiming at visitor assistance are generally based on applications centralized in a hand-held unit. Regardless of whether such applications are rich- or thin-clients, users usually interact with it by focusing their attention on a small touch screen and controlling the application by tapping in it with a stylus while holding the device with the other free hand.

Alternative interaction models have been proposed to support visitors, in which mobile hand-held units are substituted by or combined with other interaction means, such as head-worn displays, RFID tags, or public displays.

Touring Machine

The Touring Machine [Feiner 97] prototype assists users who are interested in the Columbia University campus, overlaying information about items of interest in their vicinity. Information is presented and manipulated on a combination of a head-tracked, see-through, head-worn, 3D display, and an untracked, opaque, hand-held, 2D display with stylus and track-pad.

As the user looks around the campus, his see-through head-worn display overlays textual labels on campus buildings. At the top of the display is a menu of choices. When selected, each of these choices sends a URL to a web browser running on the hand-held computer. The browser then presents, depending on the choice, information about the campus, the user's current location, a list of departments, and a list of buildings.

Rememberer

The Exploratorium, an interactive science museum in San Francisco, was the test-bed for a series of experiments with a ubiquitous computing application (part of the Cooltown project) aimed at enhancing visits to a museum. Initially, it was a typical Cooltown application: each exhibit had a set of associated web pages and point-of-information station broadcasting beacons with URLs pointing to these pages; users were carrying a PDA with which they picked up beacons and automatically presented (in a web browser) information and suggestions related to that exhibit.

However, system evaluation reported user complaints about the interference of the application in their activity. This led the research team shift to a simpler approach, based on one of the functionalities provided in the previous prototype: a remembering tool helping visitors build a record of their experiences, which they can consult after their visit. The usage of the Rememberer tool [Fleck 02] has two moments: during the visit, when users select objects they want to record and refer to later; and after the visit, when users access a set of web pages containing their visit record (a list of exhibit names in the order visited). Before starting, users visit a special base-station exhibit where a pseudonym is assigned to them. For remembering a specific exhibit, users bring an RFID tag (some credit card shaped and some mounted in watches) within 10 cm of the point of information reader, registering the exhibit under the user's pseudonym and briefly lighting up an LED. Moreover, to make the record more specific to the user's personal experiences of the exhibits, some exhibits were equipped with cameras which captured snapshots of visitors when they triggered Rememberer with the RFID tag. After the visit, visitors were given a URL where they could later view the visit record and associated pictures.

The SUPIE airport assistant

The SUPIE airport assistant prototype [Stahl 05] aims at supporting navigation and shopping tasks in an airport scenario. The navigation assistant, running on a PDA, allows the user to query information on points of interest within a three-dimensional navigational map. The user may formulate requests using combined speech and stylus input. The navigation assistant finds a route from the starting point to the target and creates situated presentations for each public display along the route to the target, which gives incremental micro-navigational route instructions guiding users to the next display until reaching their goal.

The shopping assistant provides product information and personalized advertisements to the user and is composed of: an RFID-technology based infrastructure of readers and labelled products to sense implicit user interactions, such as picking up a product from the shelf or putting it into the shopping cart; a ceiling-mounted steerable projector used to highlight products in the shelf; public displays; and a tablet PC-equipped shopping cart. Upon entering the shop, the user picks up a shopping cart and logs in to the system by a RFID customer card. As the user interacts with real products on the shelf, their actions are recognized by a RFID reader and sent as events to the application. In response, the assistant pro-actively serves product information to the user, either on the tablet PC mounted at the shopping cart, or on any nearby public display. The steerable spotlight helps the user to find certain products on the shelf, by guiding their attention to the product in question.

2.1.4 Discussion

Most of the systems here described, specially context-aware mobile applications and universal interaction systems, were designed for running on a PDA. Although PDAs excel in assisting users with work-related tasks (e.g., agenda and contacts management, note taking, etc.), they may hinder people visiting some new environment. As reported on the Rememberer study, PDAs may distract users and interfere with their activities, because users have to concentrate on the application and have to use both hands. The interaction model proposed by universal interaction systems, namely those which generate user interfaces on-the-fly based on service descriptions, is even more inadequate to visiting experiences, because it implies an additional effort from users, by requiring them to understand the semantics of interaction (the generated user interface is just a graphical counterpart of the service syntactics). Moreover, most of the systems described in this section require previous training from users.

All of these are critical aspects in the scenario I am dealing with and the reason why my approach rather explores distributed interaction with simple devices embedded in the physical environment or with personal devices requiring little or no attention to be operated. A system like Rememberer is closer to my approach, by assisting visitors while letting them focus on their activities.

2.2 Activity-centered computing

Activity-centered computing has been recognized as a fundamental concern in providing optimal and distraction-free user experiences (see section 1.2.1). Placing human activity at the core of a software system has been the major driver of several research projects. This section describes the most relevant approaches to activity-centered computing, either in desktop applications or in ubiquitous computing systems.

2.2.1 Desktop management systems

Although desktop screens have been improving in terms of resolution and dimensions, the virtual space available for organizing ongoing activities (applications, documents, etc.) is never enough, specially when multiple activities are being carried out simultaneously. Several solutions have been proposed to deal with this issue, all of them employing some sort of aggregation that encompasses applications, documents, and coordination items supporting an activity. For example, the Rooms system [Henderson 86] allows users to collect application and document windows that are part of the same activity into screen-sized rooms, one for each activity. Users can then navigate between rooms as they change their focus between activities. The Task Gallery [Robertson 00] is another example, a kind of 3D version of Rooms, where the rooms are laid out along a 3D hallway, with the current room on the wall at the end of the hall. When users want to switch from one activity to another, they just have to navigate through the 3D hallway and enter the desired room.

Similar to the above cited projects, the Kimura system [MacIntyre 01] aggregates multiple related documents, tools, and exchanged messages as a single "working context". It separates the user's desktop into two parts: the focal display on the desktop monitor, and the peripheral displays projected on the office walls. The focal display is used for the current working context, while peripheral displays are used to visualize background activities as a montage of images garnered from the activity logs, helping the user remind past actions and keeping him or her aware of updates to those working contexts.

Finally, the UMEA system [Kaptelinin 03] represents a class of desktop management systems that aggregate applications, documents, and other items into work projects based on interaction histories. Whenever the user creates and selects a project in UMEA as the current activity, the system starts to monitor user interactions (e.g., opening a document, printing a file, sending an e-mail, etc.) and associate it to the current project. The UMEA interface allows users anytime to browse their projects, manage sub-tasks, set project deadlines in a calendar, move interaction histories from one project to another (e.g., in the case the system associates an interaction to the wrong project), and open respective documents, URLs, or folders.

2.2.2 Task mobility

Many users need to migrate their computing tasks from one computer to another. For example, most would like to be working on a set of applications and documents at the office and, when going home, moving them to their laptop, and keeping all their state (open documents, window and cursor positions, etc.), all this with minimal effort. One way of doing this is by redirecting a windowing environment to different computer displays – also know as teleporting [Richardson 94]. Assuming that the computer to which the user moves implements the teleporting protocol, all the user has to do is to login to the windowing server. The user's computing environment of the last session (maybe at another computer) is teleported to the new location, retaining all the applications and documents in the state they were left.

Project Aura [Sousa 02] proposes another way of achieving task mobility, by capturing user intent (e.g., through sensors or explicitly entered by the user) and mapping it into a task corresponding to a set of abstract services. These services are further concretized by the environment infrastructure (a desktop computer, a laptop, or a PDA) providing continuous support to user tasks regardless of the environment in which the user is. The Aura of each user represents the set of services required to accomplish a task or activity and allows the user to move from environment to environment while keeping the task in execution with the resources available in that environment. Unlike teleporting, which requires homogeneous environments, Aura allows for tasks to be executed in heterogeneous environments provided the required services are available (e.g., editing a document in a laptop and continuing the same task in a PDA).

[Christensen 02] reported a pervasive computing system developed for supporting collaborative activities within health-care settings. Nurses, physicians, etc., are equipped with a PDA or use public displays in the patient rooms for carrying out their routine activities. Task mobility is achieved by using the same principles employed by Aura, i.e., user activities are described as an abstract composition of applications which are instantiated in each computing environment the user is in. Unlike Aura, however, some attention is paid to collaborative activities. For example, an activity snapshot can be handed over to other users when there is a turn shift.

2.2.3 Collaborative tools

The field of computer-supported cooperative work has produced a lot of activity-centered computing research. Many approaches have been proposed for improving collaborative activities management, by shifting the focus on the several applications used to carry out collaborative work to the central unifying concept of activity. An example of such approach is the ActivityExplorer [Muller 04], where the basic block for collaboration is a shared object. Shared objects (e.g., message, chat, file, folder, screen-shot, or to-do item) hold one piece of persistent information, and they define a list of people who have access to that content. Users are allowed to combine and aggregate heterogeneous shared objects into structured collections as their collaboration proceeds. A set of related, shared objects is called an activity thread, representing the context of a collaborative activity. The activity is carried out by inserting and editing shared objects. As collaboration proceeds, users may include new members, or exclude old members from selected shared resources in the activity thread.

Bellotti et al. [Bellotti 03] present the concept of *thrask* – threaded task-centric collection – and the Taskmaster system as solutions for the problems of common e-mail management tools that do not give enough relevance to the role of e-mail messages in task and project management. In the thrask model, e-mail messages (individual or message threads), files, and links can represent tasks. The Taskmaster automatically groups together in a thrask any related incoming messages (replies in a thread, with any attendant files or links) upon analyzing message data. Taskmaster lists thrasks in its main pane together with incoming new (non-reply) messages, which appear as single-item thrasks at the bottom of the list, rather like an email tool's inbox. The two other Taskmaster panes present thrask content (messages, attachments, and links) and the content of individual selected items. Just as e-mail message content can quickly be previewed, so can the contents of the other types of items such as web pages, spreadsheets, presentations, and documents. In this way, Taskmaster feels less like a classic application and more like a general task-management environment, handling a variety of types of media.

2.2.4 Discussion

Almost all the systems described in this section are, generally speaking, activity management systems. These systems were designed to assist users in effortless activity switching or resuming, within a single computing environment or, in the case of task mobility systems, between different, possibly heterogeneous computing environments. A kind of activity-aware layer is added between the operating system and user applications. This acts like a substitute for a file explorer or an application menu in the document- or application-centered perspectives. Furthermore, an activity is always executed in the same environment or, at most, migrates from environment to environment, but always running on a single device at a time.

In my work, the environment where activities are carried out goes far beyond the personal computer: it is composed of multiple heterogeneous devices, with each device assigned to specific actions within one or more activities. Instead of using the concept of activity as an aggregation of applications and documents, I use it as a way to communicate the user interaction model to users and to support integration of user interactions with multiple devices within an activity.

2.3 Activity modelling

As an abstraction, the concept of activity has to be concretized into a machine-understandable manner. Research on activity models has been done with several different goals. This section describes those which are more closely related to my work, namely: a) defining the behavior of autonomous entities; and b) representing the way humans perform an activity.

2.3.1 Directing autonomous entities

By autonomous entities, I mean artificially intelligent entities, such as software agents or robots, that are able to behave autonomously, e.g., making decisions or reacting to events in their environment. However, the level of autonomy of such entities is always restricted by some human-defined plan or program describing, for instance, how the entity behaves in face of specific situations or what parameters are taken into account when making a decision.

Most agent-based systems or robotics approaches [Simmons 98, Paolucci 99, Wise 00, Look 03] adopt a solution for planning the behavior of autonomous entities that can be summarized as follows. A global plan is defined to achieve some high-level goal. The plan is decomposed in tasks and sub-tasks. Some tasks – simple tasks – correspond to executable actions (run by some agent) while others are complex tasks made of several simple tasks. In these systems, there is a component responsible for executing the plan, by decomposing complex tasks into simple ones and delegating simple tasks to agents for execution. Tasks are usually described as a set of parameters, inputs, outcomes, resources needed, and constraints that should hold either before, after or during the execution of the task. What the task actually does (e.g., computation, physical action, etc.) is not part of the plan, but internal to the agent itself. Along the execution of a plan, a monitoring component verifies that task constraints are being satisfied. Exception handling and event-reaction mechanisms are also common in these systems.

2.3.2 Representing human activities

This category of activity modelling can be divided into user interface models and models of work. In both models, human activities are oriented by some goal and are decomposed into smaller units of abstraction (tasks, operations, etc.), which are realized by humans in order to accomplish that goal.

In user interface models – most known as task models –, human activities are seen from the point of view of user interface designers. These task models describe how activities can be performed by means of user interface widgets in an application (e.g., entering password in a dialog box, clicking some button, etc.). Such models are used when people have to understand how a system works or in usability evaluation where the purpose is to evaluate how well the system task model supports users [Paternò 01]. Several approaches to task modelling have been developed, namely: GOMS [Card 83], which structures tasks as a set of goals, operators as elementary acts, methods to achieve those goals through operators, and selection rules to indicate when to use a method instead of another one; User Action Notation [Hartson 92], which, in one part, describes task decomposition and temporal relationships among tasks, and in the other part, associates each basic task to interface feedback and system state; or ConcurTaskTrees [Paternò 99], which uses a graphical notation to describe task decomposition, relationships among tasks (enabling, concurrency, optionality, etc.), task allocation, and resources required by tasks.

Another approach for representing human activity is by focusing on organizational processes. Workflow systems [Medina-Mora 92, Georgakopoulos 95, Schäl 96] have been widely used in organizations as ways of formalizing work procedures. Workflow specification languages use rules, constraints, and/or graphical constructs to describe task structure, the ordering and synchronization of tasks in a workflow, task attributes that describe the tasks (e.g., task duration or priority) and the roles to perform them, and exception handling. Bardram [Bardram 97], influenced by the work of Suchman on situated actions [Suchman 87] as an alternative to the rigidness of workflow, proposed an activity model based on plans. Plans are here seen as a resource for the work rather than determining its course. An activity model is thus composed of action templates (the actual plan) that give an orientation to work but that, when instantiated, can be adjusted to the actual context of activity.

2.3.3 Discussion

Generally, robotics and agent-based approaches model activity at a rather detailed level, specifying step-by-step the path of actions. Although these rigid approaches make sense with robots or agents, who have no free will, an alternative is needed for specifying human activity, which is rather unpredictable. Even within organizational processes, where the boundaries of human activity are more defined, workflow models have been criticized for their lack of flexibility [Suchman 87, Bardram 97]. Task models for user interfaces, though effective in helping software designers for communicating and supporting further development efforts, focus on the system's view of activity rather than on the user's perspective.

The type of activities my work is aimed at have often an informal nature and thus cannot be modelled using a workflow-like approach. The activity model I propose in this work is comparable to the more flexible approach followed by Bardram, whose model provides a kind of guide for activity that is later instantiated and carried out at the will of users. However, though employing an approach that differs from most of the above presented models, my model shares with them the general structure and many concepts.

2.4 Applications of Activity Theory

Activity Theory has been in the last two decades recognized as a useful conceptual framework for understanding human activity and for supporting the design of and reflection about information systems. It has influenced multiple fields, such as human-computer interfaces [Bødker 91b, Nardi 96, Bannon 91], computer-supported cooperative work [Kuutti 91b, Bardram 97, Christensen 02], and information systems design and development [Kuutti 91a, Bødker 91a]. This new research direction has been fostered by the recognition that the traditional rationalistic thinking is insufficient as a theoretical basis for systems design [Bødker 91a, Bannon 91]. However, as a conceptual framework rather than a formal theory in itself, most researchers have some difficulties in concretizing Activity Theory concepts in their results [Constantine 06]. As Kaptelinin et al. state, "these [Activity Theory's] principles help orient thought and research, but they are somewhat abstract when it comes to the actual business of working on a design or performing an evaluation" [Kaptelinin 99].

Therefore, some efforts have been carried out to leverage the richness of Activity Theory's conceptual framework and developing tools that make Activity Theory more practicable and, at the end, more useful. The first attempt is the Activity Checklist [Kaptelinin 99], a guide to the specific points a researcher or a practitioner should consider when trying to understand the context in which an activity takes place, and consequently, in which a tool will be or is used. The checklist is divided into two versions: one targeted at system design and another to system evaluation (43 and 37 items respectively). Some items are shared between the two versions, although rephrased to meet the respective analysis goal. The checklist structure is composed of four sections reflecting the main concepts of Activity Theory: means and ends; social and physical aspects of the environment; learning, cognition, and articulation; and development. A set of sample questions for each section is also provided.

Recently, Duignan et al. proposed the Activity Interview [Duignan 06] as an enhancement to the Activity Checklist. Their claim that the Checklist is too long and still abstract and not accessible to practitioners unfamiliar with Activity Theory resulted in a shorter, though comprehensive, set of questions made of everyday language, accessible to any human-computer interaction analyst. The checklist's design and evaluation versions were merged and some duplicate or closely related items were amalgamated. Most of the Activity Theory jargon was removed, in order to enable fluid discussion, and so that interviewers can quickly determine the relevance of the various items to the current interview context, rather than having to first think about the meaning of some concept. Both the Activity Checklist and Interview authors warn that, despite Activity Theory's holistic approach, these artifacts should not be used in isolation, but should be complemented with other research methods that are more effective in uncovering other system dimensions.

Finally, the most concrete and pragmatic application of Activity Theory is Constantine's Activity Modelling [Constantine 06], a method and set of notations for capturing and representing salient information regarding activities that is most relevant to interaction design. Activity Modelling provides a link between Activity Theory and usage-centered design [Constantine 99], a model-driven process for user interface and interaction design primarily focused on usage rather than on users *per se*. Specifically, Activity Modelling adds to usage-centered design the concise constructs for representing the contextual or collective aspects of work it lacked. The main additions to the usage-centered design process are: the Activity Model, that defines and describes activities and their interrelationships; the Role Profile, that describes the user roles, is modified to connect roles explicitly to the activities within which the roles are embedded; and the Task Model, that is elaborated to incorporate actions[1] in relation to artifacts and

[1]In Activity Modelling, the term *task* is distinguished from *action* in that the former is relative to activity in interaction with the system being analyzed, while the latter is relative to the activity in general, without implying interactivity.

participants and to connect task cases – particular tasks in a user role – explicitly to activities.

2.4.1 Discussion

Activity Theory provides a deep understanding of the mental processes related to human activity and of the contextual, historical, and social forces influencing it. Many of the above cited works used Activity Theory as a conceptual framework for reasoning about human activity, but had no concrete application of its concepts. In my work, I adopted a pragmatic approach, by focusing my work on a subset of the theory. Given that my work just proposes a generic interaction model and infrastructure for ubiquitous computing solutions, the design of the actual support to specific activities can employ analysis tools such as the Activity Checklist or the Activity Interview or design methods such as Activity Modeling. The Activity Checklist and Interview are particularly comprehensive regarding the whole range of Activity Theory concepts and should thus be recommended for activity-based design.

2.5 Distributed user interaction in ubiquitous computing environments

This work assumes that interaction with the ubiquitous computing environment in a public place is distributed (in space and in interaction mean), i.e., visitors interact throughout the space with many different devices, either personal or local, in order to carry out their activities, instead of concentrating interaction in a single device, as is usually the case of desktop or mobile computing. As introduced in section 1.3.2, the major challenge of distributed interaction in a ubiquitous computing environment is making the user feel that all the interactions, whatever device is used, are integrated and part of the same activity. Most ubiquitous computing research has assumed that interaction is made through a single device (usually a PDA or a mobile phone), but some work has explored the fundamental problems of distributed interaction. The next subsections present what has been done namely in design frameworks and implementations of distributed interaction.

2.5.1 Design frameworks

Bellotti et al. [Bellotti 02] identified a set of typical design challenges to user interfaces for ubiquitous computing environments. In contrast to familiar graphical user interface mechanisms such as cursors, windows, menus, etc., that provide pre-packaged solutions to nowadays basic user interaction problems, ubiquitous computing is still far from such answers. Drawing on the theoretical framework of human-human interaction in social science, they identify five main issues: address – directing communication to a system; attention – establishing that the system is attending; action – defining what is to be done with the system; alignment – monitoring system response; and accident – avoiding or recovering from errors or misunderstandings. For each issue, corresponding familiar graphical user interface answers are presented and detailed challenges and problems are discussed along with related ubiquitous computing research. Although not attempting to provide a systematic framework for dealing with those issues, the work of Bellotti et al. is a source for reflection and discussion about the design of distributed interaction in ubiquitous computing.

An additional contribution to distributed interaction design has been made by Kray et al. [Kray 04], who propose a multi-dimensional model and a set of design issues to characterize and reflect on user interaction in ubiquitous computing. Their design model comprises: a user dimension, which distinguishes interaction based on the location of users (collocated to distributed), group size, and degree of collaboration; device properties, such as degree of privacy (public or private), shareability, affordance, or sensing (passive or active); and an interaction dimension distinguishing direct (directly manipulating a physical object) from indirect interaction (manipulating a virtual representation). Regarding the design issues, these are grouped into three main categories: management issues, e.g., user identification, device assignment, device control (when multiple users share the same device), or load-limit; technical issues, e.g., synchronization (when multiple devices are used for the same input/output), device interference, device coverage, or system performance; and social issues, e.g., social rules or privacy.

Again, this is a tool mainly for reflection and discussion as no concrete solutions to the aforementioned issues are presented.

2.5.2 Implementations

Gaia [Román 02a] and BEACH [Tandler 01], both further detailed in section 2.7.2, are ubiquitous computing infrastructures that deal to some extent with the problems of distributed user interaction. Gaia is presented as a meta-operating system for managing ubiquitous computing environments. The Gaia architecture is layered into several levels of abstraction. The application-level functionality is based on a framework derived from the Model-View-Controller [Krasner 88] pattern. It is composed of five components: model, presentation, controller, input sensor, and coordinator. The fact that input and presentation are separated from the application logic enables device-independent applications and, therefore, scenarios in which users interact with the same application through many different devices within a ubiquitous computing environment. One of the paradigmatic examples of this interaction framework is the calendar application that, when running in an ubiquitous computing-enabled office, may use a plasma display to present the appointments for the week, a handheld to display the appointments for the day, and may use a desktop PC to enter data. However, the same calendar running in a vehicle may use the sound system to broadcast information about the next appointment, and use an input sensor based on speech recognition to query the calendar or to enter and delete data.

The BEACH infrastructure aims at supporting synchronous document-based collaboration with heterogeneous devices in a *roomware* environment. Like Gaia, the application framework is based on the Model-View Controller pattern. However, BEACH does not offer the same flexibility provided by Gaia, which is not restricted to collaborative scenarios and allows for application bridging (e.g., automatically coupling the outcome of an interaction with some device to the input of another device).

In contrast to Gaia and BEACH, which explore a generic architecture for distributed interaction in ubiquitous computing environments, the Command Post of the Future (CPoF) [Myers 02] is a system targeted at military environments. Military staff sat in front of large displays showing maps and military unit information can use multiple modalities of input (speech, laser pointing, gestures) and presentation (wall displays, PDAs, mobile phones) to monitor and control a military force in the field. Users can switch between interaction modalities as required by conditions in the field (e.g., switching from speech to handwriting because a loud airplane is passing overhead). CPoF is distinguishable from other multi-modal approaches due to its distributed character, allowing the use of many different devices during a session (e.g., a PDA, a laser pointer, and speech recognition attached to a wall display).

2.5.3 Discussion

Similarly to Gaia and BEACH, and in contrast to CPoF, my work aims at a generic architecture for supporting distributed interaction in ubiquitous computing environments and thus leveraging the advantages that flexible device usage brings to visitors. In order to achieve this goal, I use the same principles of abstraction and separation of concerns, i.e., separating application logic from input and presentation, that those infrastructures implement.

The design frameworks presented in section 2.5.1 are a valuable source of reflection and offer guidelines upon which I base the distributed interaction model I propose in chapter 3. I particularly focus on the *address-attention-action-alignment*, and to a lesser extent, *accident* dimensions of the design framework suggested by Bellotti et al.

2.6 End-user programming and configuration

In a near future, ubiquitous computing environments are going to be as common as electrical systems are. Nowadays, when we want to shutdown or turn on electrical supply in some room or floor in a building, we do not need an electrical engineer or a technician: we just know there is a switch board where we can easily perform these actions. Likewise, ubiquitous computing environments need similar tools so that common users, without ubiquitous computing expertise, can configure or program an environment, e.g., at their home or office. This section presents some efforts that have been dedicated to end-user programming and configuration toolkits for ubiquitous computing environments. These toolkits can be classified from many perspectives. The perspective I chose for presenting and comparing these approaches with mine is whether the toolkit is used for programmed context-aware behavior or for configuration of system components. In the former case, the user programs the system to perform some action as a result of some context change, whereas in the latter case, the user configures existing system components to build new behavior out of it.

2.6.1 Context-aware programming

The general model for end-user context-aware programming is the following: the user identifies a set of situations that are going to trigger some noticeable user-defined action in the environment. Situations are detected through context sensors available in the environment, whereas actions are performed by actuators attached to ubiquitous computing devices. The approaches described below follow quite close this model, differing mainly in the programming paradigm and user interface abstractions.

SiteView [Beckmann 03] and iCAP [Dey 06] are both end-user programming toolkits targeted at defining rules for context-aware system behavior. The main difference between both approaches is the user-interface: SiteView is based on a tangible user interface, while iCAP is a visual programming tool. In SiteView, users create rules by manipulating tangible interactors, physical objects that represent sensed conditions such as *monday* or *rain* and automated actions such as *light on* or *increase temperature*. Interactors are manipulated on top of a table divided into two areas: the condition composer, on top of which conditional interactors are placed to form a conjunction; and a world-in-miniature, a small-scale representation of the physical environment, where the user places interactors representing a spatially defined action. Additionally, two displays provide some feedback about the programmed behavior: an environment display showing photographs of what the environment will look like when a rule is activated; and a rules display showing the rule (close to human natural language) as it is created and other rules applicable for the given set of conditions.

iCAP shares the conceptual model with SiteView, aggregating all the programming tasks into a single graphical user interface, divided into two main areas: a tabbed window on the left that is the repository for user-defined elements (people, artifacts, etc.); and, on the right, a rules area where elements from the repository can be dragged to construct a conditional rule. The rules area is split into two sub-areas: one for conditions and the other for actions. For example, for specifying a rule "when John is in the kitchen then play music", the users drags from the repository to the conditions area the elements representing John and the kitchen location and drags to the actions area the element representing the music player. After a rule has been defined, it can be tested using the iCAP rules engine.

The aCAPpella system [Dey 04] allows for the creation of context-aware programs by exploring the programming by demonstration paradigm. In this system, the user first demonstrates a context-dependent behavior that includes both a set of conditions and an associated action, then tells the system which portions of the demonstration are relevant and repetitively trains aCAPpella on this behavior over time by giving multiple examples. Once trained, aCAPpella performs the demonstrated action whenever it detects the demonstrated conditions. The aCAPpella system is composed of: a set of sensors (video camera, microphone, RFID readers, etc.), with which the user demonstration is captured; a graphical user interface with which the relevant sensor data and time interval for the behavior are selected by the user; and a machine learning system that is used for training aCAPpella recognition capabilities.

The CAMP [Truong 04] tool is aimed at end-user programming of context-aware activity capture and playback. CAMP is based on an infrastructure for capture and later access that uses sensors (generally cameras and microphones) deployed in, e.g, a domestic environment. The captured content can later or simultaneously be played back for remembering or following a particular activity (for example, remembering last week-end party or monitoring a baby while she is playing). The CAMP graphical user interface is based on a magnetic poetry metaphor: users combine individual words to form a poem in almost natural language. A CAMP poem instructs the system about the context and object of a capture. Each word represents one of the following: who (e.g., "I", "everyone", "Joe", etc.); what (e.g., "conversation", "picture", etc.); where (e.g., "kitchen", "everywhere", etc.); when (e.g., "always", "A.M.", "hour"); and other general concepts ("record", "display", "view", "keep", etc.). For example, the poem "when Jim Jane and Billy talk record and remember for 20 minute" would be interpreted by the system as "whenever a microphone senses Jim, Jane, or Billy's voice, start audio recording and keep the record for 20 minutes".

2.6.2 System configuration

System configuration is here understood as the combination of elements of the ubiquitous computing infrastructure in order to produce a desired behavior, much like writing a script with a series of commands or assembling components to obtain a new artifact.

Alfred [Gajos 02] is a multi-modal macro recorder for a ubiquitous computing infrastructure. Upon a user's request, the system begins recording all of his commands (primarily spoken) and, when the recording is done, the user assigns one or more spoken names (the macro name) to the recorded sequence. The user can also add hardware triggers (e.g., pressing a button) to the macro. A command sequence can be made of, e.g., "turn on the main lights", "open the drapes", or "turn on my desk lamp" spoken commands. Whenever the user says the macro name (e.g., "Good morning"), the infrastructure executes the corresponding command sequence.

The approach employed in the AutoHAN infrastructure [Blackwell 01] for programming home appliances is the usage of the Media Cubes programming language. A Media Cube is literally a cube made of wood, containing a variety of sensors, transducers, and a microprocessor, and communicating with home appliances through infra-red ports. Each Media Cube represents an appliance function category, with each face representing a specific function. Media Cubes are used much like generic remote controls. Media Cubes programming is done by invoking, over the infra-red channel, specific functions in an appliance by holding a cube or a set of connected cubes against the front of that appliance. Media Cubes users can employ two different programming paradigms: the ontological paradigm, in which each cube represent a category of functions (e.g., a cube for events like "go"/"stop" or "on"/"off", another cube for indexes like TV channels or messages in an answering machine, etc.); and the linguistic paradigm, in which cubes represent words in a language and can be combined to form commands (e.g., combining a "remove" face of the "list" cube with the "3" face of the "index" cube to remove the third message of the answering machine).

The Jigsaw Editor [Humble 03] also provides end-users with high-level abstractions for system configuration. This tool is based on a simple jigsaw pieces assembling metaphor, with each jigsaw piece representing a ubiquitous computing device (e.g., a display, a grocery alarm in the fridge, a doorbell, etc.). The editor's graphical user interface is composed of two panels: a list of available components (shown as jigsaw pieces) and an editing canvas. Jigsaw pieces can be dragged and dropped into the editing canvas. By combining the outputs and inputs of the available devices, i.e., coupling in a left to right fashion compatible jigsaw pieces, the users can configure the environment to produce a desired behavior. When a user places two devices close to each other, they snap together if the property types for the underlying devices input and output match. For example, by combining, in this strict order, jigsaw pieces representing a doorbell, a web camera and a PDA would produce an information flow resulting in signaling the web camera to take a picture whenever the doorbell is pushed and directing the picture to the PDA.

The Equip Component Toolkit (ECT) [Greenhalgh 04] aims at facilitating the job of ubiquitous computing application designers and administrators by providing graphical tools which provide various representations of the running environment, plus facilities for monitoring, configuration, scripting and learning by example. The toolkit supports distributed applications running over multiple hosts by the creation, configuration and interconnection of software components and components representing physical devices and sensors. Toolkit users manipulate graphical representations of components and interconnect them to build new applications. This approach is similar to Jigsaw's, though employing metaphors with a lower level of abstraction – based on component-based models – for representing system components.

2.6.3 Discussion

In the solutions to end-user programming and configuration described above, the target users are mainly the actual users of a ubiquitous computing environment. Although my approach is aimed at public space administrators, the need for user interface abstractions adequate to users with no technical expertise is the same. The tool I propose can be classified as context-aware system configuration, because it allows administrators to assemble a set of input/output devices and software components available in an environment to support specific actions and activities, with the possibility of taking context into consideration for constraining the availability of actions and activities to visitors. Therefore, context is not used by administrators to define the outcome of actions or activities, as is the case of the systems described in section 2.6.1, but is rather used to enable or disable specific actions or activities.

Although all the tools described above can be used to support specific activities, users do not think primarily of activities when interacting with the tool, but rather think of input/output devices or sensors in a certain situation. The metaphors, symbols and concepts employed in these tools are essentially for abstracting devices or device functionality rather than higher level concerns. Even in CAMP, where users do not have to necessarily think of specific devices, they think of situations rather than of activities. In my approach, system administrators primarily define the activities and actions visitors are going to be provided with and then configure the interaction devices and software components supporting those activities and actions and the context in which they are going to be enabled.

2.7 Ubiquitous computing infrastructures

Ubiquitous computing research has produced along the years many infrastructures characterized by specific architectures and programming models. Some research efforts have been targeted at extending the personal computer's operating system model to ubiquitous computing environments, while others have just provided basic middleware services upon which applications are developed or configured.

2.7.1 Speakeasy

The Speakeasy project [Newman 02] explores the concept of recombinant computing as a solution to deal with the diversity and heterogeneity of resources users may find in their physical environment. The recombinant computing approach is similar to the philosophies of encapsulation and reuse behind component-based frameworks. However, these concepts are extended in the way that entities are exposed to end-user to be used and configured in highly dynamic and ad-hoc ways, such that components can interact and interoperate with each other even though they were built with no prior knowledge of one another. This requires the use of semantically-neutral interfaces (e.g., for data exchange, discovery, user interaction, representation of contextual information), so that entities interact in a very basic way without needing to rewrite code. Developers thus focus on writing individual components and ensuring their compliance to the agreed interfaces, leaving the task of application composition for users.

It is assumed that users have knowledge about the resources they encounter in the environment and that they are able to manipulate and compose them in order to fulfill their needs. Basically, the Speakeasy system allows users to discover, through a web browser, devices available in the environment and combine them to achieve some functionality (e.g., combine a presentation file with a presentation viewer and a projector control). Besides this basic mode of operation, for some recurrent activities, templates are provided (or created by users themselves) for the composition and manipulation of available resources. For example, a "give a presentation" template would be instantiated in an environment using the available resources matching the required resource types. This approach is similar to the philosophy of the Aura project (see section 2.2.2).

2.7.2 Gaia

Gaia [Román 02b] is a component-based middleware infrastructure for ubiquitous computing applications offering a meta-operating system – GaiaOS – that provides a generic computational environment for ubiquitous computing and an application model that defines a standard mechanism to build applications. Gaia converts physical spaces and their ubiquitous computing devices into a programmable computing system called "active space". GaiaOS is analogous to traditional operating systems. GaiaOS manages the computational resources within a physical space and mediates interaction between the active space and ubiquitous computing applications. GaiaOS is composed of two main parts: the Unified Object Bus and the Kernel. The Unified Object Bus provides tools to manipulate uniformly heterogeneous components running in the system. The GaiaOS Kernel includes essential services implementing the core functionality of the system (entity discovery, component repository, event distribution, naming, data storage and manipulation, and security).

Furthermore, Gaia proposes an application model for ubiquitous computing scenarios. This application model is based on the traditional Model-View-Controller paradigm, augmented with extensions to account for the characteristics of ubiquitous computing environments. The Model-Presentation-Adapter-Controller-Coordinator (MPACC) model is the implementation of the application's central structure, which normally consists of data and a programmable interface to manipulate the data. The Presentation is the physical externalization of the model that allows users to perceive it visually, by voice, etc. The Controller exports mechanisms to modify the state of the model, either through user input or through sources of context that can affect the application. The Adapter is the component responsible for adapting the format of the model to the characteristics of output devices. The Coordinator is a meta-level component that manages the application composition and applies adaptation policies, based on functional and non-functional properties.

2.7.3 Interactive Workspaces

The general goal of the Interactive Workspaces [Johanson 02] project is similar to Gaia's: constructing a higher level operating system, called iROS, for ubiquitous computing environments – particularly collaborative workspace environments. However, their focus is on augmenting widely deployed legacy applications such as web browsers and desktop productivity tools with collaborative behaviors and the ability to handle multi-modal input. They also focus on the use of PDAs as remote controllers for logical or physical entities in the workspace.

The iROS meta-operating system is composed of three main sub-systems: the Event Heap, a tuple-space-derived mechanism by which entities in the ubiquitous computing infrastructure communicate indirectly; the Data Heap, a data store facilitating data migration among heterogeneous computing platforms, including data conversion between different formats; and iCrafter, a system for service advertisement and invocation, along with a user interface generator for services that generates the interface according to device characteristics.

2.7.4 BEACH

BEACH [Tandler 01] provides the software infrastructure for ubiquitous computing meeting-room environments supporting synchronous collaboration with many different devices, building on shared documents accessible concurrently via multiple devices. The BEACH architecture is organized in four layers (module, generic, model, and core layers) and specific models separating concerns within each layer. The module layer provides tailored functionality to distinct tasks. It can be used to extend the functionality defined by the generic layer, which contains generic components that provide the basic functionality for teamwork. The model layer specifies the basic structure for the two top layers by defining interfaces to the system model (documents, tools, etc.). Finally, the core layer mainly provides multi-user event handling, device and sensor management, and a shared-object space to allow distributed access to the model from multiple computers.

The model layer comprises interfaces for five models: document, tool, user interface, interaction, and physical environment. The document model defines the base classes and functionality of all objects that can be part of a document. The tool model describes the elements that are directly attached to the user interface, providing additional functionality to the documents. The user interface model is needed to define an alternative user interface concept suitable for different devices. Furthermore, multiple-computer devices require that the user interface elements are part of the shared-object space. This enables user interface elements to be distributed among several computers. The physical model is the representation of relevant parts of the "real" world. To be able to support different styles of interaction the interaction model specifies how different interaction styles can be defined.

BEACH application developers comply to the above described architecture by reusing components from the generic or module layers and implementing additional functionality structured according to the BEACH models.

2.7.5 ACCORD

The ACCORD project [Humble 03] aims to develop a framework that allows a user-centered reconfiguration of domestic ubiquitous computing environments. The user-oriented framework exploits a component model based on JavaBeans [Sun 03] and shares bean properties across a distributed data-space. The shared data-space allows real-world devices to make information about the nature of the physical environment digitally available. Devices can use this data-space to become aware of their context, represent this contextual information to other devices, and to make this manifest in the physical world. Each component of the system is implemented as a digital/physical transformer, i.e., a component that take digital/physical events and transforms them into physical/digital representations. The properties of each transformer are published in the shared data-space.

ACCORD developers just have to concentrate on developing transformers for their devices and publishing information about their characteristics and state into the shared data-space, leaving the task of application configuration to users, very much like the Speakeasy approach (see section 2.7.1). In this case, ACCORD provides users with a simple editing metaphor, based on the notion of assembling simple jig-saw pieces, described in section 2.6.2.

2.7.6 Discussion

All the above described systems provide some sort of generic computational infrastructure for developing pervasive applications. Some infrastructures, like Speakeasy and ACCORD, provide only a minimal support to application development and usage: only a distributed communication middleware and generic device interfaces are available and the applications are eventually composed out of the available devices by end users. Others, especially Gaia and Interactive Workspace, are more ambitious and reproduce a kind of conventional operating system for ubiquitous computing environments. While the infrastructure I propose aims at reducing the efforts of both end users and developers and, as such, cannot be as simple as Speakeasy and ACCORD are, it is not intended to be an operating system. It rather shares some of the goals of BEACH, i.e., providing a generic programming model and a set of middleware services and reusable components. But the major distinctive characteristic of the infrastructure I propose is its activity-centered conceptual model that is present in all the levels of abstraction of the system: from the lower level interfacing with ubiquitous computing devices to the highest level of user interaction modelling; and this latter aspect is unique, because no other infrastructure clearly proposes a generic user interaction model, being the decision left to application developers.

2.8 Integration with personal resources

The support to an activity performed by occasional visitors requires the combination of several components from the personal and the local domain that cannot be provided solely by the ubiquitous computing infrastructure or by the personal domain. Therefore, local and personal domains have to integrate with each other in order to achieve both a user-centered support and a thorough exploration of the local resources (see section 1.3.5). This section describes some of the research that has been done to make the several facets of the personal domain available to ubiquitous computing infrastructures, so that the integration between both becomes possible.

The VADE project [José 03] introduces the concept of Value ADded Environment as an administrative and physical domain where the locally available computing facilities can be combined with the personal environment of visiting users. The common scenario of a personal environment is a mobile portal that, after the user enters a VADE, provides functionality that corresponds to the dynamic combination of predefined preferences, currently active applications, current user context and locally available services and applications. The local applications are dynamically integrated into the personal environment by means of Web Services for Remote Portlets (WSRP) [OASIS 03].

The User Virtual Space concept [Hess 02], which has been used in the Gaia project (see section 2.7.2), is an abstraction to represent the user's data, tasks, and devices in a ubiquitous computing environment. The User Virtual Space is implemented as a set of personal file servers which may be running in different hosts. The local environment has a mount server which obtains from a user's handheld device the mount points corresponding to the User Virtual Space. The mount server then merges such mount points and presents data to the user, in a file browser, as if it came from a single source.

The Personal Server [Want 02] is a mobile device designed to enable interaction with a user's personal data through the surrounding computing infrastructure (screens, keyboards, etc. of nearby computers with a short-range wireless link). Whenever users need to access their personal data, they just have to find a nearby display and keyboard with wireless connection. In addition, the personal server contains enough storage and processing capabilities to serve a user's mobile computing and storage needs. Therefore, the personal server is not only a storage device but is able to host applications inside. The personal server model addresses two major problems associated with mobile information access: the inherent difficulty of using small user interfaces on handheld devices, and the limited access to personal digital information afforded by public access points.

The Pendle [Villar 05] proposes a mixed-initiative approach for user interaction with ubiquitous computing environments. It assumes the existence of an environment that provides a set of adaptive services and a personalized wearable device – the Pendle, realized as a pendant on a ribbon worn around the neck – that serves as mediator between user and the proactive environment. In the mixed-initiative approach, two interaction modes are considered: implicit and explicit. In the implicit mode, the Pendle sends to the local environment, every five seconds, the user profile (under the user's control), giving the environment access to personal data for adaptation of its services (e.g., playing music according to the user's preferences). In the explicit mode, it provides a set of controls with which the user can modify or override the behavior of the adaptive services (e.g., controlling music sound volume).

The Mobile Service Toolkit (MST) [Toye 05] is a client-server framework exploring mobile phones as a means for accessing site-specific services. The MST client performs three primary functions: connection establishment to an MST server, after recognizing on-site 2D codes representing Bluetooth or IP addresses; data entry and display for supporting interaction with site-specific services; and personal information management. Regarding this latter aspect, the MST client manages a repository of personal information (owner name, address, phone number, etc.) that can be supplied on server request and after owner's disclosure, to MST servers.

2.8.1 Discussion

The integration between the local infrastructure and the personal information domain requires necessarily some compromise between both parties, generally in the form of agreed protocols for communication and privacy. The less widespread and generic these protocols are, the more obstacles to successful integration show up, particularly if integration requires specific software or hardware on the user side. Requiring visitors to carry additional cumbersome devices or to install specific software in their personal devices is contrary to a lightweight visiting experience.

Almost all the above described integration mechanisms require specific hardware or software from users. In some cases, these requirements do not hinder a visiting experience (e.g., the Pendle or the User Virtual Space). However, all these mechanisms are based in proprietary integration protocols that would have to strive to impose itself in the marketplace. The exception is the VADE system, which is based on standard web technologies. However, it is restricted to the usage of a mobile browser, which is not the user-friendliest approach for supporting a visiting experience.

Although not aiming to propose a solution for the local-personal integration issue, this work explores simple integration mechanisms based on widespread technologies, much in the line of the user interaction approach, also based on simple, widespread devices. My approach for integration, though somehow limited in the depth of information that can be shared with the local environment, has the advantage of not requiring from visitors extraordinary software or devices.

2.9 Summary

This chapter provided an overview of the multi-disciplinary character of my work, covering subjects from ubiquitous computing infrastructures to user interaction issues, with particular emphasis on activity-centered research. In every covered subject, I demonstrate how my work relates to others and also what makes it a distinct research effort. In summary, my contribution is unique in that it explores the usage of widespread, simple interaction technologies for supporting visitors to public spaces in a way that requires little or no attention from users, which is strengthened by an activity-centered perspective of user interaction, in which activity is the main concept supporting the user interaction model and the integration of user interactions with multiple devices within an activity. I propose a ubiquitous computing infrastructure based on an activity-centered conceptual model that is present in all the levels of abstraction of the system: from the lower level interfacing with ubiquitous computing devices to the highest level of user interaction modelling. This infrastructure is complemented by a management tool allowing administrators to assemble and configure a set of ubiquitous computing devices and software components available in an environment to support specific actions and activities.

Chapter 3

Modelling activities and user interaction

Modelling human activity and user interaction in ubiquitous computing environments is a challenging matter, one for which a definitive, comprehensive solution is very hard to achieve (see sections 1.3.1 and 1.3.2). Two decisive characteristics of the scenarios this work deals with, i.e., visiting experiences to probably unknown environments, determine my approach to activity and user interaction modelling: on the one hand, human behavior is unpredictable and profoundly influenced by context in which it occurs (personal situation, physical environment, other people, etc.); and, on the other hand, occasional users of a ubiquitous computing system are not willing to invest time in learning a system that probably is not going to be used again and thus demand straightforward, simple user interfaces. Therefore, my research on activity and user interaction model is driven by the fundamental principle of simplicity. When designing for an activity, overall simplicity in features play a prominent role [Abowd 00]. A simple conceptual model is not just an advantage for end users – the simpler the provided model, the faster and easier they will be using the system – but brings also benefits for application developers and system administrators, in terms of development and administration costs. This model is thus aimed at embracing all the major concerns of a ubiquitous computing infrastructure: from system architecture and administration to user interaction.

I decided to ground my research on theoretical frameworks of human activity, namely the Activity Theory framework, because it provides an agreed set of terms to describe human activity and concepts that drive the construction and evolution of systems that intend to support human activity [Constantine 06]. I also followed a simplistic analysis of user interaction, reducing it to elementary concepts and, most importantly, assuming the usage of simple, everyday devices.

This chapter is divided into three main sections. The first two sections intend to describe the characteristics that have to be considered when modelling human activity and user interaction in ubiquitous computing environments and to present the theoretical foundation for the ActivitySpot conceptual model that is finally described in section 3.3 and that bases the remainder of my work.

3.1 Modelling activity

The activities that can be carried out in a ubiquitous computing environment have to be modelled according to several characteristics, such as activity structure, physical space, tools, context, among others. This section begins with an overview of Activity Theory, which provides the theoretical foundation for activity modelling in this work. The section further relates activity to several dimensions and closes with a discussion of the implications of Activity Theory and all the activity modelling dimensions in the ActivitySpot conceptual model.

3.1.1 Activity Theory

Activity Theory [Leontiev 78, Vygotsky 78, Leontiev 81, Wertsch 81, Engeström 87, Engeström 99b] was chosen as the background for this work, among several theoretical frameworks for studying human activity, produced mainly by the fields of psychology and philosophy (e.g, situated action [Suchman 87], cognitive science [Norman 88, Sierhuis 97], or Actor-Network Theory [Law 99]). Activity Theory originated in the former Soviet Union's school of psychology, mainly through work done by Vygostsky and Leontiev. Although its origins come mainly from the first half of the twentieth century, Activity Theory is still evolving. Much recent thinking in Activity Theory has been influenced by the work of Engeström [Engeström 87, Engeström 99a], who continues to develop the theory, methods and practice of what has come to be known as Cultural-Historical Activity Theory. The following paragraphs expose the major features of Activity Theory, supported by examples of everyday human activities.

Activity is understood as the unit of life that is mediated by mental reflection. The real function of this unit is to orient the subject in the world of objects [Leontiev 81]. Activity Theory is fundamentally based on the notions that human activity: a) is analyzed at different levels; b) is goal-oriented; c) is mediated by tools; d) has an historical, cultural, and genetical basis; e) is based on social interaction; and f) is assimilated by individuals through internalization. These fundamental concepts are further developed.

Human activity can be analyzed at three distinct levels: *activities*, at the uppermost level, are distinguished on the basis of their motive and the object toward which they are oriented; *actions* are distinguished on the basis of their goals; and, finally, *operations*, on the basis of the conditions under which they are carried out (see figure 3.1). For example, an activity motivated by food is composed of several goal-oriented actions (e.g., collecting ingredients, following a recipe, etc.) and operations which vary in function of conditions (e.g., the action of collecting ingredients may be composed of different possible operations, such as going to the kitchen-garden, picking vegetables, taking a piece of meat from the fridge, etc.).

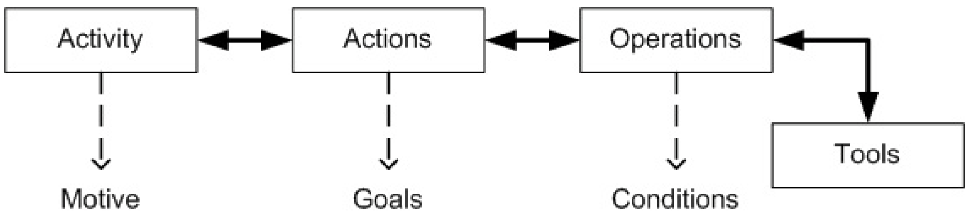


Figure 3.1: Activity Theory model

However, the structure of activity is not just a simple decomposition of activity into lower level elements. An important feature of the activity structure in Activity Theory is that it is relatively content free in the sense it is not tied to a particular set of structurally defined steps [Wertsch 81]. This functional approach, contrasting to the traditional western structural approach, means that a link (e.g., an action or an operation) in an activity can be replaced by another one that is functionally equivalent, i.e., that fulfills the same motive or goal. For example, it may happen that someone does not have time to cook and decides to buy food at some take away restaurant. In this case, a set of actions was replaced by another that realizes the same activity motivated by food. An activity may thus be carried out in a variety of ways by employing different goals (with their associated actions) under different conditions (with their associated operations).

As human activity is inherently goal-oriented, the process of goal formation is an important psychological issue. The subject performing an activity has to select the goals that are going to contribute to achieve the results necessary to satisfy its motive. In this process, the consciousness of the subject regarding motive and goals within an activity is dynamic. It may happen that an activity loses the motive that inspired it, whereupon it is converted into an action (with its own goal) that may have a quite different relation to the world, i.e., implement a different activity. For example, one subject performing an activity motivated by food at the week-end can think of it as an action during the week, just because she is a restaurant manager, for whom procuring ingredients and cooking are not actions motivated by food but are now motivated by profit. Cooking may not even be an action for the restaurant manager, but become an operation, because it does not contribute directly to realize the motive of profit.

Conversely, an action can acquire an independent, energizing force and become an activity in its own right. One of the actions involved in an activity in one situation may be considered to be an entire activity in another situation. For example, procuring ingredients is an action for a domestic cook preparing the dinner for her family, but can become an activity in its own if we think of the procurement department in a restaurant chain.

An action may also be part of different activities. The goal of the action is exactly the same, but it serves a different motive. For example, following a recipe can serve the motive of food when cooking for the family or the motive of fame and professional recognition in a cook world contest.

Apart from its intentional aspect (what must be done), the action has its operational aspect (how it can be done), which is defined not by the goal itself, but by the objective circumstances under which it is carried out (operations). If we imagine a case in which the goal remains the same and the conditions under which it is given change, then only the operational composition of the action changes. Finally, an action can be transformed into a means of attaining a goal, i.e., into an operation capable of supporting the accomplishment of one or more actions. This generally happens when an individual ceases to execute an action consciously, doing it "mechanically" (e.g., shifting gears in a car, in which case the action would afterwards be changing the car's speed) or delegating it to a machine.

A particular characteristic of human activity is that it is mediated by tools – psychological (e.g., speech, arithmetics, mental plans, etc.) or physical (e.g., an axe, a pen, a computer, etc.). Each operation may require some tool to be executed. When a tool executes an operation automatically, it allows the individual to concentrate on actions and activities, freeing her/him from low-level efforts. This is known in Activity Theory as the process of internalization. It is generally the fate of operations that, sooner or later, they become a function of a tool. Tools are both resources for, and products of, human activity, i.e., humans tend to adapt existing tools, or to build new tools, in order to better support an activity.

The process of externalization, opposed to internalization, occurs when an operation regains the subject's consciousness. For example, the driving instructor has to externalize driving operations in order to teach them or, when a breakdown occurs during the execution of an operation, the subject has to externalize it in order to understand what went wrong.

An activity naturally evolves over time, influenced by cultural and historical forces. Furthermore, humans constantly search better means to attain their motives, learning with errors, and correcting the way by which their activities are performed. Another aspect of human activity is that it is a result of social interaction, i.e., beyond supporting specific activities, social interaction influences activity and allows people to share their experiences and therefore spreading the knowledge about how to perform activities (e.g., when adults teach children).

Although Activity Theory provides a rich framework for explaining human activity from different perspectives, its concrete application has not been abundant. Even if some research claims to use or to be influenced by Activity Theory, it is not clear how the results have been actually shaped by Activity Theory (see section 2.4). This can be explained by its lack of formality – Activity Theory comprises a collection of concepts and categories for communicating about activity coupled to diverse assertions about the nature of human activity that are largely untested [Constantine 06].

3.1.2 Physical space

The close relationship between activity and physical space has been recognized by architects [Alexander 77], anthropologists [Hall 66], and computer scientists [Harrison 96, Koile 03]. In this work, I am particularly interested in activities that have a strong relationship with physical space. The physical space determines what activities can or cannot be performed and influences the way they are performed. When modelling an activity, details such as the plan of the area or building and the location of furniture, equipment, and interaction devices are of crucial importance. For example, actions that are usually executed at the initial steps of the activity should primarily be supported in entrance zones (e.g., a lobby or hall); a zone of the building that is usually crowded may not be a proper place for allowing voice-based interaction; the lack of wide, clean walls may hinder the installation of large public displays. Therefore, the activity model must include details about the locations in the physical space where actions and activities are to be executed and the type of interactions with the ubiquitous computing environment that are allowed in each location.

3.1.3 Tools

A computer-supported activity may require several computer tools. The computer tool is here understood as any hardware/software computing artifact: a physical device, an information service, a digital document, etc. Tools required by an activity may come from different domains: personal, local, and global. Personal tools provide an individualized experience to users, either by using their own personal devices or by introducing a personal dimension to the logical/physical artifacts they use (e.g., through identification, personalized adaptation of contents, personalization of device color and shape, etc.). Personal tools are either brought to the local environment by the visitor (e.g., mobile phone, PDA, etc.), made available as remote services (e.g., a URL to a personal profile service), or be temporarily lent to visitors (e.g., an RFID tag for user location purposes).

Local tools are those which are owned or managed by the local infrastructure, providing visitors with access to the physical environment or representations of it (e.g., a light control service, a local map, etc.) or with access to scarce or not easily portable computing means (e.g., a public display, a printer, etc.).

Global tools are those which are not managed by the local infrastructure nor associated to the visitor. Global tools provide impersonal information access in any place and to any user, generally through an Internet service (e.g., a portuguese-english translation web service or a weather forecast web site).

The model of a particular activity must thus include a reference to the local, personal, and global tools that are required for carrying out the activity. The actual local and global tools needed for an activity are likely known in advance, either because they are managed by the provider of the support to the activity (local tools) or are previously contracted (global tools). However, personal tools are unknown and dynamic by nature – occasional visitors come and go and no assertions about the type and ownership of personal tools can be made. Anyway, the activity model must at least make an abstract reference to the tools that the local infrastructure expects the visitor to own.

3.1.4 Context

Activities depend on context at two distinct situations: when a visitor has just entered the ubiquitous computing environment and the system has to check which activities are available or to infer which activity may the visitor be interested to accomplish; and when the activity is being performed, influencing how it is unrolled.

When activity selection is explicitly done by visitors, context factors are used to filter the set of possible activities (e.g., visiting the exhibits in a museum can only be done within a certain time range or ice skating can occur only within adequate weather conditions). Implicit activity inference occurs when the ubiquitous computing infrastructure pro-actively selects the activity the visitor is likely to initiate. This inference is partially based on context factors, such as time, location (e.g., if the visitor entered the hospital by the patient entrance, she is likely coming for a consultation), number of grouped visitors (e.g., two adults and children entering a museum are likely coming for a regular visit), among others. In either case – explicit or implicit activity inference –, the activity model includes expected values for the different context factors that can be measured in the environment.

During activity execution, context plays also an important role, by influencing the availability and outcome of actions. For example, when visiting a shopping mall, checking the restaurant's menu makes sense only within a certain time range or when looking for a particular product, results can be ordered by proximity. The role of context lays here at two different levels of the activity model: at the action level, where context influences the availability of actions; and at the operational level, where context influences the outcome of actions and the tools that can be used to execute the action (e.g., voice interaction cannot be used in a momentarily noisy environment).

3.1.5 Personalization

An important aspect of the support to activities is the level of personalization offered to visitors. As it happens with context, personalization can occur at the moment of activity selection (e.g., a museum visitor interested in sacred art will likely be offered a sacred art tour by the system) or during activity execution, by influencing the way actions are performed and their outcome (e.g., elderly or disabled people have to perform an activity possibly in a very different manner).

Personalization can also be useful in the case a visitor returns to a previously visited the environment. The same previous activity can be suggested to the visitor or some preference can be taken into account during activity execution. For example, a conference participant that repeatedly selected a vegetarian menu in the last conference held in the same place will by default be assigned the same menu when registering at the conference.

Personalization is inherently attached to privacy issues. Visitors have to be aware that the infrastructure is gathering information about themselves and have to be given control over personal information disclosure [Abowd 00].

3.1.6 State and history

Interruptions between activities are quite common, even if, in a visit scenario, the set of possibly overlapping activities is smaller than in office or home scenarios. Hence, an activity model must cater for activity interruption and further resumption. This requires the representation of activity state at the moment of interruption. Activity state should be modelled in a way that is both machine-processable (for resumption by the system) and easily convertible to human language (for resumption by visitors). Activity state includes both actions and respective operations executed until the moment of interruption, as well as additional information such as the state of any local object (physical or virtual) used by the visitor.

The same mechanisms used for activity state management can also support activity history, another important aspect of the activity model. Activity history can not only be used for computing the activity state or helping the user remember the point at which the activity was left, but also for reviewing activity performance (for example, remembering how the visit to the cultural center was or what relevant actions were done during a conference), for remembering a previous activity when visiting the same environment after some time, or for sharing our own experience with other people.

3.1.7 Discussion

Most of the concepts proposed by Activity Theory, though not immediately evident for application in ubiquitous computing, are extremely valuable as a foundation for activity modelling in the context of this work. The levelled approach for analyzing human activity (activity, actions, and operations), although not radically different from other approaches (e.g., cognitive science), provides a simple and yet essential model for structuring activities. Conscious and “mechanical” behavior are clearly separated, helping in the identification of the elements of activity that should place little or no mental overhead to visitors. For example, system designers should model activities being aware that the only tolerable effort from visitors should be in quickly identifying available activities and actions. Everything regarding operations should be easily executed by visitors, requiring only little or peripheral attention. Therefore, modelling activities with the visitor's motives and goals in mind, helps in identifying the right terminology to use in the support to those activities. The ultimate result of this approach is gradually contributing to the internalization of activity, i.e., helping visitors to concentrate on the higher level concerns of activity without thinking of or being disturbed by the usage of tools, achieving a performance with the minimum required level of attention. This is the ultimate goal of ubiquitous computing, as stated by Weiser [Weiser 91]: whenever people learn something sufficiently well, they cease to be aware of it.

From the systems engineering point of view, Activity Theory interestingly supports the principles of modularity and reuse. The same activity can be carried out by executing a diversity of actions and operations, depending on context and on personal characteristics. Furthermore, the same action or operation can be employed in different activities, contributing to achieve respectively different goals and motives.

The concept of tool mediation helps understanding the role of technology in the way activities are carried out. Tools are not only fundamental in the support to human activity but are as well drivers in the creation of new activities that were impossible before [Wertsch 81]. Likewise, ubiquitous computing tools are not only effective means for supporting human activities but are decisive in the creation of new forms of carrying out activities or may even originate entirely new activities. Furthermore, tools evolve over time, as a result of both technological improvements and evolution of praxis (people introduce changes to tools that improve activity performance). A model of activity should thus facilitate evolution and adaptability to technological change.

The dimensions of activity state and history described in section 3.1.6 can be interrelated with the concepts of externalization and cultural and historical development of Activity Theory. A kind of externalization process is occurring when the execution of activity is tracked by a ubiquitous computing infrastructure in order to keep the activity state. What is done consciously (from the level of activity) or unconsciously (to the level of operations) by the visitor is transformed into an explicit representation by the system. This system externalization of activity can also lead to externalization by the visitor himself, when he accesses the activity history or is given hints for resuming activity. The activity history is also a means of contributing to the cultural and historical development of activity: on the one hand, cultural development can occur when activity history is handed to other people in order to share experiences; on the other hand, the transmission and remembrance of activity history contributes to the historical development of activity by supporting collective learning and the introduction of improvement into the activity praxis.

In summary, I propose an activity model that, yet simple, is able to support the comprehensive analysis of Activity Theory and all the relevant dimensions of an activity. For such a model to be possible, I have to deal with a compromise between simplicity on one side and comprehensiveness and completeness on the other side. The solution I propose is further described in section 3.3.

3.2 Modelling user interaction

This work assumes that visitors to a ubiquitous computing environment are going to find a possibly vast array of local devices, tending to be blended with the environment or, at least, easy to use. These devices may be as diverse as the specificities of each activity (e.g., cameras, displays, microphones, speakers, LEDs, buttons, joysticks, RFID readers, etc.). Moreover, the support to an activity may allow visitors to employ their own personal interaction devices (e.g., a mobile phone). Therefore, a designer of the support to an activity must take into account these disparate media and integrate them into a unifying concept of interaction, easying both visitors' and developers' task, by making the interaction transparent, despite the differences that exist between each type of medium. Another challenge is to deal with the many different interaction devices the same person may use within the course of an activity and to make that person feel that all interactions, whatever device is used, are integrated and all part of the same activity.

These user interaction issues implicitly include many others, which are common to ubiquitous computing and general human-computer interface issues. The following subsection provides an overview on previous research on user interaction design, which I next base on to discuss the user interaction problems my work aims to deal with.

3.2.1 User interaction design models

According to Norman [Norman 88], user interaction designers must recognize the importance of leading users to clearly discern the effect of their input in the system and to control it. He proposes the Gulfs of Execution and Evaluation as a model of the interaction between users and computer tools. The Gulf of Execution corresponds to the mental process from goal formation to actual action execution: users begin to identify system capabilities that enable them to achieve their goal, then specify which operations must be executed, and finally execute them. During the Gulf of Execution, users should easily know what actions are possible, determine mapping from intention to physical movement, and perform the action.

After action execution comes the Gulf of Evaluation, when the user compares the system outcome with her goals. The action (and respective operations) executed on the system produces changes on its physical or virtual state. After perceiving the system state, the user interprets it and evaluates whether the outcome has realized her goals. During the Gulf of Evaluation, users should easily tell what state is the system in, determine mapping from system state to interpretation, and tell if the system is in desired state.

Norman summarizes this with a set of requirements:

- good conceptual model – the user's model of interaction with the system, i.e., how the user perceives interaction with the system, is coherent with the design model, i.e., what interaction with the system actually is;
- visibility – the user can easily tell the state of the system and the alternatives for action;
- good mappings – the user easily determines the relationships between actions and

results, between the controls and their effects, and between the system state and what is visible;

- feedback – the user receives full and continuous feedback about the results of actions.

Bellotti et al. (see section 2.5.1), following Norman's work, propose a design model for novel interaction mechanisms, such as ubiquitous computing, with a focus on the joint accomplishments of the user and the system that are necessary to complete interaction. This approach is based on the assumption that users interact, for example, with ubiquitous computing systems, much like they interact with other humans, managing accomplishments such as addressing, attending to, or politely ignoring one another. They identify five basic issues for interaction design, inspired by Norman's stages of execution and evaluation, but with the emphasis being on communication rather than on cognition. Each issue and respective detailed challenges are here enumerated:

- address – what mechanisms does the user employ to address the system, how to disambiguate signal-to-noise, how to disambiguate intended target system, how to not address the system;
- attention – making users know whether and when the system is attending to them, how to direct feedback to the zone of user attention;
- action – how users effect a meaningful action, control its extent and possibly specify a target for that action (corresponding to Norman's Gulf of Execution);
- alignment – monitoring system response (corresponding to Norman's Gulf of Evaluation);
- accident – avoiding or recovering from errors or misunderstandings, how to control or cancel system action in progress, how to disambiguate what to undo in time.

Kray et al. (see section 2.5.1) identify a set of design issues of multi-user multi-device interfaces in general, which are as well common issues in ubiquitous computing environments. Those issues are divided into management, technical, and social ones.

I focus here on management and technical issues, given that social issues go beyond the aim of the interaction model I am considering here, which is concentrated on the problems of interaction itself. Management issues regard:

- user registration, identification, and verification as users enter and leave a ubiquitous computing environment, in order to assess the current needs and capabilities of the system and to provide information about the user population characteristics (e.g., user location, group membership, preferences, etc.);
- device assignment, because multiple users may use the same device to access several services at the one time, and a single service may require the use of multiple devices to operate at another time; furthermore, device assignment may be influenced by user location and surrounding conditions;
- device control, when users compete for the same non-shareable device or which one user does not want to share; device control conflicts may also arise when the type of interactions in a shared device are not compatible (e.g., viewing a movie with a high sound volume combined with Internet surfing);
- load-limit, determined by the ratio between the number of users and the number of available devices.

Among the technical issues, I point out:

- synchronization, when interaction simultaneously spans multiple input or output devices;
- interference, when multiple output devices are simultaneously presenting contents to different users (e.g., interference between public audio channels in a small space);
- coverage, as users can only interact and communicate if they are in range of an adequate device; the level of coverage depends on device properties, physical placement, and density of users;
- system performance, specially when the overall complexity of an environment grows through increased multiple user and multiple device interactions.

3.2.2 Discussion

User interaction design for ubiquitous computing environments, beyond involving issues that are common to traditional desktop interfaces, brings many other issues that turn it into a much more complex design problem. In order to better understand what is involved in modelling user interaction in the ubiquitous computing scenarios my work is targeted at, I analyze here the issues and concepts enumerated above in the context of activities performed by occasional visitors.

People visiting some place for the first time have no idea of the ubiquitous computing support to the activity they intend to perform there. They may have some cues learnt from past experiences in similar places (e.g., past visits to airports taught them that public displays provide flight information), but still they do not know in detail the available ubiquitous computing means and the activities and actions they allow them to perform. For example, an airport visitor may not know that the system is able to automatically check her in and guide her to the departure gate after recognizing her frequent flyer card. Moreover, since it is the first time the visitor is interacting with the local system, it will be unable to identify further user interactions with other devices until some initialization action is performed (e.g., voice-based interaction requires previous training for later identification of the user's voice pattern). This *initial contact* problem aggregates the above mentioned issues such as user registration, addressing the system, or the first step of the Gulf of Execution. Presenting the visitor with a conceptual model that is coherent with the user's model of the intended activity is key in facilitating initial contact. However, a good conceptual model alone is not enough: interaction devices are distributed all over the physical space and most of their capabilities are not self-explainable. For example, desktop user interfaces rely on a screen to provide users with indications on how to use the system. However, a ubiquitous computing system may not have a screen at all. Like in a desktop computer without a screen, where the mouse and the keyboard cannot convey usage information to users, microphones, video cameras and other input devices in a ubiquitous computing system must rely on external means to convey usage information. Visitors have therefore to be provided throughout the physical space with visible, comprehensive anchors to system usage, e.g., in the form of public displays, posters, flyers, and so on.

Fundamental in capturing visitors' attention for system usage are some principles for immediate usability identified in [Kules 03]: immediate attraction – using the most attractive content to demonstrate the system and invite usage; immediate learning – support zero-trial learning; immediate engagement – encourage users to immediately interact with content by providing immediate reward and avoid interruption; and immediate disengagement – when a user departs, immediately prepare the system for the next visitor. Although these principles were defined for interactive kiosks, most of them can be applied to the context of this work.

After visitors become aware of the support to their activity, they begin the actual interaction. Some issues identified in the previous sub-section come into play at this phase: determining if the system is *attending* to the user; *addressing* the system; and then, *acting*, i.e., carrying out the remaining steps of the Gulf of Execution.

Given that the same interaction device may be used by multiple visitors for multiple different purposes, a significant issue is visitor and action *identification*. Visitor identification requires previous registration at the system and as many identifiers as many interaction device types are available (e.g., face recognition for video camera-captured gestures, phone number for SMS interaction, etc.). Action identification depends on the content of the actual interaction (e.g., different voice commands for different actions), on the context of activity (e.g., the same gesture may trigger a different action for different activities) or on the activity history (e.g., the same SMS command may trigger different actions if executed at the beginning or later in the activity).

After identifying the input author and intention, the system has to process it and generate a response or, at least, some feedback. From the user perspective, this leads to the *alignment* or Gulf of Evaluation issues. But before the user is presented with a response or feedback, other issues arise. Multiple output devices can be available for providing a response (e.g., public display or speakers for responding to a gesture interaction). A decision has to be made regarding the output device that is best suited to the visitor context and to the response content. For example, choosing a public display may be the wisest decision if the environment is noisy or if the response has to contain an image to be more effective. Moreover, after selecting an appropriate device, comes the issue of content adaptation to device characteristics. This is not a problem when the device belongs to the local environment, because device characteristics can be anticipated at development time, but it becomes a difficult issue when the response is sent to some visitor's output device. Personal device characteristics cannot be anticipated for many output modalities (e.g., screen resolution, supported sound formats, etc.), unless they are communicated to the local infrastructure after an initialization procedure.

Management and technical issues, such as device assignment and control, load-limit, or device coverage are also important at this stage of interaction. When choosing a suitable output device, the infrastructure must take into account whether the device is already being used by other visitors or if the device coverage is enough for reaching the user's attention.

Finally, one of the most important issues is making the person feel that all interactions are part of the same activity, regardless of the interaction devices they are employing. Key aspects in achieving this are not only a coherent interaction model but also coherent symbology on signage displayed throughout the physical space and coherent language and signs on the interaction content. Providing regularly visitors with information about their activity state through different interaction means is also determinant in transmitting that feeling of integrated interactions.

In summary, supporting occasional interaction with a ubiquitous computing environment faces many issues that can be divided into communicative – making the visitor know that there is something there to support her and enabling immediate usage – and technical ones – accessing personal data, identifying interactions, or selecting appropriate device for response. The interaction model I propose in the next section represents mainly technical issues, though its simplicity can contribute to facilitate communication of the system capabilities to visitors.

3.3 The ActivitySpot conceptual framework

The ActivitySpot conceptual framework aims to provide a simple model of activity and user interaction for visitors to a ubiquitous computing environment. This framework intends to support designers and developers of new ubiquitous computing solutions to human activities and is concretized by the ActivitySpot infrastructure described in chapter 4. The following sub-section describes the ActivitySpot conceptual framework and is followed by a discussion about how the framework deals with the issues identified in sections 3.1 and 3.2.

3.3.1 Concepts

Activity Theory provides a rich and comprehensive framework for analyzing human activity. However, the need for a simple and objective approach for modelling human activities in ubiquitous computing environments results in a partial application of Activity Theory. Among its concepts, I am particularly interested in the different levels of analysis of an activity – *activities*, actions and *operations* –, in the flexibility of the activity structure – opposed to a rigid set or sequence of actions – and in the reuse at the level of actions and operations that can be respectively part of multiple activities and actions. Ubiquitous computing devices are seen as tools used for the execution of operations. A ubiquitous computing device can be employed in the execution of operations that are part of different actions.

The diagram in figure 3.2 illustrates the application of Activity Theory to a ubiquitous computing environment – for instance, a museum – supporting two different activities: visiting, by regular, general public visitors; and inspecting, performed by security inspectors. Each activity is composed of a set of possible actions. Visitors can orient themselves, view information about some artwork, or make recommendations. Likewise, inspectors may also need to orient themselves, make recommendations, and view information about equipment. None of these actions is mandatory neither a sequence has to be followed. Some actions can be executed in both activities, with the same goal in mind (e.g., orient) or with different aims (e.g., make recommendation).

Operations include user interactions with a ubiquitous computing device, a sensor read, a web-service request, a database query, etc. Only user-facing devices are represented, which are the most visible part of operations. The same user interaction operation can be done within different actions (e.g., looking at a public display for orienting or for viewing artwork or equipment information), while other user interactions are restricted to a single action (e.g., using joystick, SMS, hardware button, or RFID card).

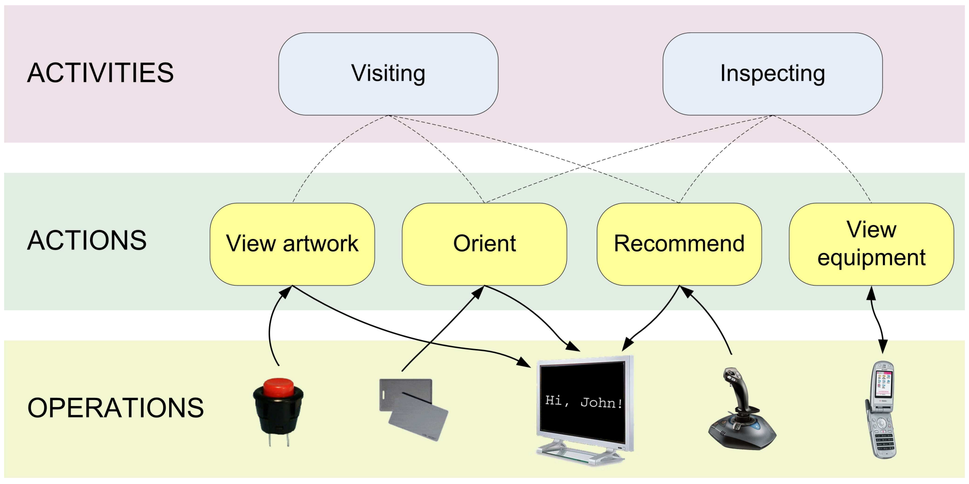


Figure 3.2: Example of an activity-centered model of a ubiquitous computing environment

The concepts of activity and action are thus part of the ActivitySpot conceptual model (see figure 3.3). Operations are omitted from the model for two reasons: 1) they are executed unconsciously by visitors and therefore are not as relevant as other concepts; and 2) as many operations correspond to user interactions, they are implicitly represented by the user interaction concepts described below. The model considers furthermore that activities or actions depend on local and personal context, either as an execution condition or as a variable influencing the outcome of an action. Some activities and actions can only be executed within certain conditions. For example, the visiting activity can only take place within a specific time schedule or the "make recommendation" action can only be executed if the visitor has previously provided his name to the system. Context can also determine the outcome of actions. For example, the outcome of the "view artwork" action may depend on the lighting level in the room, being the content format adjusted to amount of light near the public display.

Visitors are also a first class entity in the ActivitySpot model. Their identification, personal data, and preferences have to be taken into account either as a condition for activity execution or as information determining the outcome of actions, while the activity is being carried out.

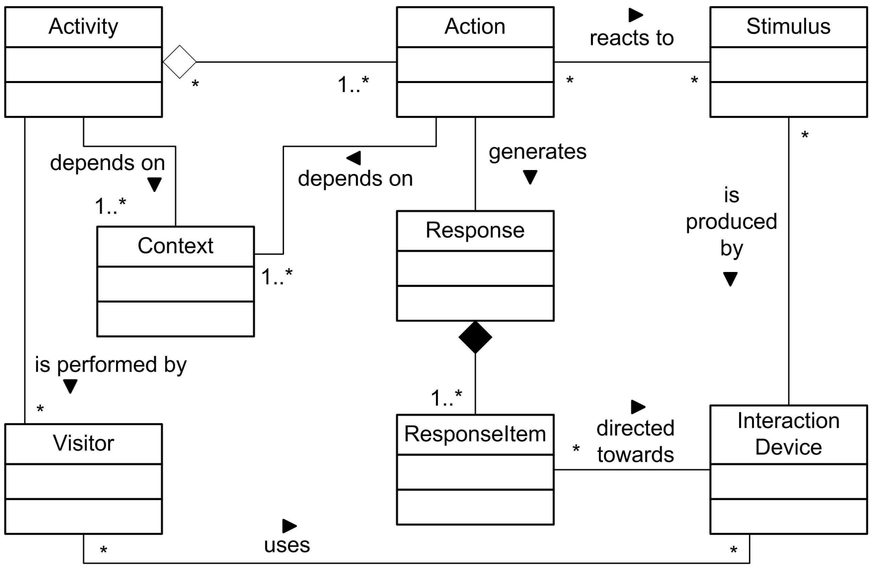


Figure 3.3: The ActivitySpot conceptual model

For modelling user interaction, I also followed an approach based on simple concepts. Given that user interaction with a ubiquitous computing system is done through multiple, heterogeneous devices and, in many cases, with little common characteristics, I reduced user interaction analysis to basic human-computer interaction concepts: stimulus and response. When a visitor uses an interaction device to provoke a stimulus to the system (e.g., handing her personal RFID card to a reader), she is doing it in order to execute some action. Therefore, the logic associated to that action is going to react to the stimulus, process it, and then generate a response to the visitor through some other device. When a response is directed to more than one device (e.g., simultaneously presenting some content in a public display and sending an SMS to the visitor's phone), it is composed of several response items.

Given that user interaction is partitioned among multiple, heterogeneous devices, it has to be decoupled from the logic supporting the activity and its respective actions, so that changes in the interaction means do not considerably affect how the support for an activity is implemented. Therefore, the interaction model I propose is informed by works such as Model-View-Controller [Krasner 88] or Model-Presentation-Controller-Coordinator [Román 02a], which are driven by the need of separating interaction concerns from logic and data.

Another distinctive characteristic of the interaction model I propose is the relation between interaction and the dimensions of time and space. According to Fitzmaurice [Fitzmaurice 95], there are time-multiplexed devices and space-multiplexed devices. The most well known example of a time-multiplexed device is the mouse. The same device is multiplexed over time to control different graphical user interface widgets. On the other hand, the space-multiplexing model is based on the idea of associating specific physical objects to specific functional aspects of the application. The objects become dedicated functional manipulators. An example of space-multiplexed device is an audio mixing console, where each slider is associated to a specific music channel. The interaction model I propose combines both concepts and corresponds to a time-space-multiplexed model. Some interaction devices can be employed for executing many different actions along the time while others can be dedicated to a specific action. In the example illustrated by figure 3.2, the button is space-multiplexed, while the public display is time-multiplexed.

3.3.2 Discussion

The ActivitySpot conceptual model does not pretend to cover all the issues identified in the previous sections. Yet, it deals with the most important ones, while still preserving the required simplicity of concepts individually and of the model as a whole.

Regarding activity modelling, the application of Activity Theory is centered on the structural aspect of human activity. Other aspects, such as goal formation, tool mediation, externalization-internalization, or cultural-historical development are not explicitly represented in the model. The focus on the activity structure, besides contributing to the simplicity of the model, by pruning out less familiar concepts, emphasizes the aspects that are closer to the common perception of activity. Presenting visitors with a model of activity that is centered on the motive for their visit and on the actions they can execute to carry out that activity provides significative advantages in terms of rapid perception of the support provided by the ubiquitous computing environment. Furthermore, the structural aspect of activity is a catalyzer for modularity and reuse. In a software engineering perspective, actions are seen as loosely-coupled units of activity with self-contained behavior, implementing a simple contract based on reactions to stimuli and generation of responses, which enables action reuse among multiple activities.

Due to its abstract nature, the application of this framework into concrete scenarios must be preceded by a field analysis of activity, its decomposition into actions and operations, and contextual elements influencing activity execution, using tools such as the Activity Checklist or the Activity Model (see section 2.4). This analysis is comparable to task analysis [Diaper 03], which is performed in cognitive science approaches to user interaction design.

By decoupling activity logic or, more precisely, operational logic from the interaction devices used by visitors, i.e., separating input, logic, and output, the ActivitySpot framework facilitates the seamless substitution of interaction possibilities as well as the introduction of new, unanticipated devices, thus supporting evolution. The ActivitySpot model makes no assumptions about interaction devices other than being able to produce stimulus and response events.

The ActivitySpot framework does not pretend to provide solutions for the user interaction issues identified in section 3.2.2. However, it provides the basic abstractions needed for representing interaction at different moments and thus hiding the lower level, device-dependent concerns. The stimulus and response concepts, their association to interaction devices and, indirectly, to visitors, can be employed during:

- the initial contact stage, namely at registration, which can be done through a stimulus to the system, identified as a registration step (for instance, a registration SMS);

- addressing and acting on the system – actions are executed by stimulating the system through available devices;

- action and visitor identification, by inspecting the type and content of a stimulus;
- alignment – responses are generated as a consequence of stimuli, taking into account that a response may be directed to multiple devices or that a choice may have to be done between multiple alternatives for output (e.g., choosing the output device that is closer to the visitor).

Furthermore, since all stimuli and responses are framed into some action and, consequently, into an activity, the model enables the communication to visitors of a feeling that all their interactions are integrated into their activity. For example, for each response, if suitable, the system can convey activity status information based on previous interactions and on further possible interactions.

Finally, the ActivitySpot model includes support for specifying activity and action execution conditions, depending either on context (e.g., location, time, light, weather, etc.) or on the visitor profile (e.g., role, preferences, etc.). Though visitors consciously disclose access to their profile or are aware of context information gathering, execution conditions are not necessarily published to visitors. This part of the model is rather targeted at system designers or administrators for configuring the support to activities. The context and visitor profile models are intentionally abstract, i.e., no assumption is made about their structure. The concretization of context and visitor profile models are left to ActivitySpot implementations.

3.4 Summary

The ActivitySpot framework intends to provide occasional visitors to public spaces with a simple, activity-centered interaction model that facilitates rapid perception of how the ubiquitous computing environment can support their activities. Furthermore, the framework intends to provide system designers and administrators with simple and yet comprehensive abstractions for developing and managing ubiquitous computing support to activities. The ActivitySpot model is informed by Activity Theory, a theoretical framework for analyzing human activity, and by several user interaction models, some of them specific to ubiquitous computing interaction. Most of the issues and concepts identified by those models and frameworks are incorporated into the ActivitySpot model or adapted to the scenario this work is targeted at, resulting in an activity and user interaction model that combines simplicity and comprehensiveness. The following two chapters describe how the ActivitySpot model is put in practice into a ubiquitous computing infrastructure – the ActivitySpot infrastructure – and in a ubiquitous computing environment modelling and development toolkit – the ActivitySpot toolkit.

Chapter 4

The ActivitySpot software framework

The realization of the conceptual model described in the previous chapter into concrete solutions for human activities in public places requires four major elements: 1) deployment of interaction devices throughout the physical space; 2) development of the logic and content for the actions supported in that place; 3) configuration of supported activities and actions; and 4) a runtime software infrastructure managing user interaction and activity execution. While the first element is out of the scope of this thesis, the remaining elements are the subject of this chapter.

This chapter describes the ActivitySpot software framework, a set of software tools and a run-time infrastructure supporting the deployment of ubiquitous computing solutions for public places. The deployment of an ActivitySpot solution comprises three phases, each served by a specific component of the ActivitySpot software framework: the development phase consists in the creation of action controllers that implement the logic associated with actions – a software library is available for facilitating the rapid development of new action controllers (see section 4.3); the configuration phase consists in the definition of which activities, actions, and interaction devices will be available at a specific environment, and their interrelations – this phase is assisted by a GUI-based authoring tool (see section 4.4); finally, in the deployment phase, the ActivitySpot infrastructure coordinates interactions with devices and manages the execution of actions within each visitor's activity (see section 4.2).

The first section of this chapter presents the ActivitySpot architecture, which is the foundation for all the software composing the ActivitySpot framework. The next three sections describe the run-time infrastructure, the software library, and the GUI-based authoring tool. In each section, requirements analysis is presented and further discussed against the proposed solution.

4.1 The ActivitySpot architecture

The ActivitySpot architecture implements the conceptual model defined in section 3.3. The main component of the ActivitySpot architecture is the Activity Manager, a user interaction and context management system that bases its decisions on a formal specification of the supported activities and devices. The class diagram in figure 4.1 depicts the general perspective over the ActivitySpot architecture.

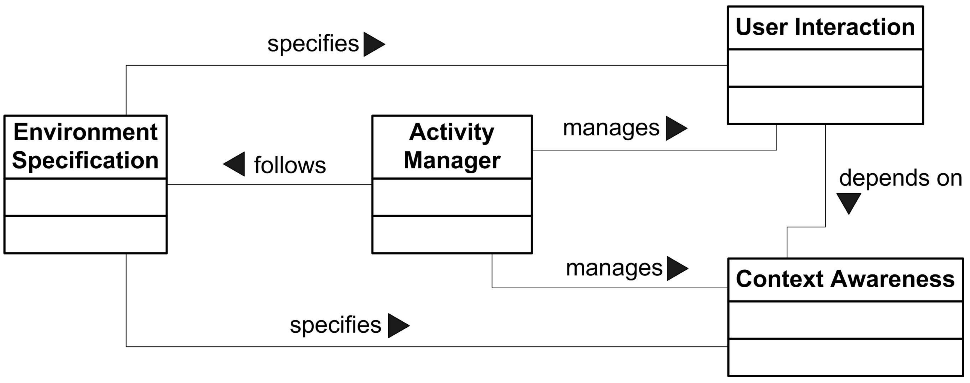


Figure 4.1: A class diagram overview of the ActivitySpot architecture

ActivitySpot is based on a local environment specification that describes the activities that are supported, actions composing it, and the devices available in the physical space for carrying out activities. Furthermore, the environment specification details what devices can be used for executing specific actions and what context dimensions determine the available activities and actions. This environment specification can be produced with the help of a GUI-based authoring tool described in 4.4. By following the environment specification, the Activity Manager knows the characteristics of the local setting, being able to manage user interaction and context-awareness.

The following sub-sections detail the architecture of environment specification, user interaction, and context.

4.1.1 Environment specification

In order to be independent of physical space and activities and thus support any activity scenario, the ActivitySpot framework is based on a generic specification format for activities, actions, and interaction devices available in an environment (see diagram 4.2). Each environment supported by ActivitySpot has a specification of: a) which actions can be executed – name, description, supported stimulus and response types, a reference to the components implementing actions (action controllers), and execution conditions; b) which activities are available – name, description, execution conditions, and references to the actions composing it; c) which local devices can be used – stimulus or response type, physical location, and references to other devices which have some physical or logical association; and d) which context dimensions can be used for specifying execution conditions.

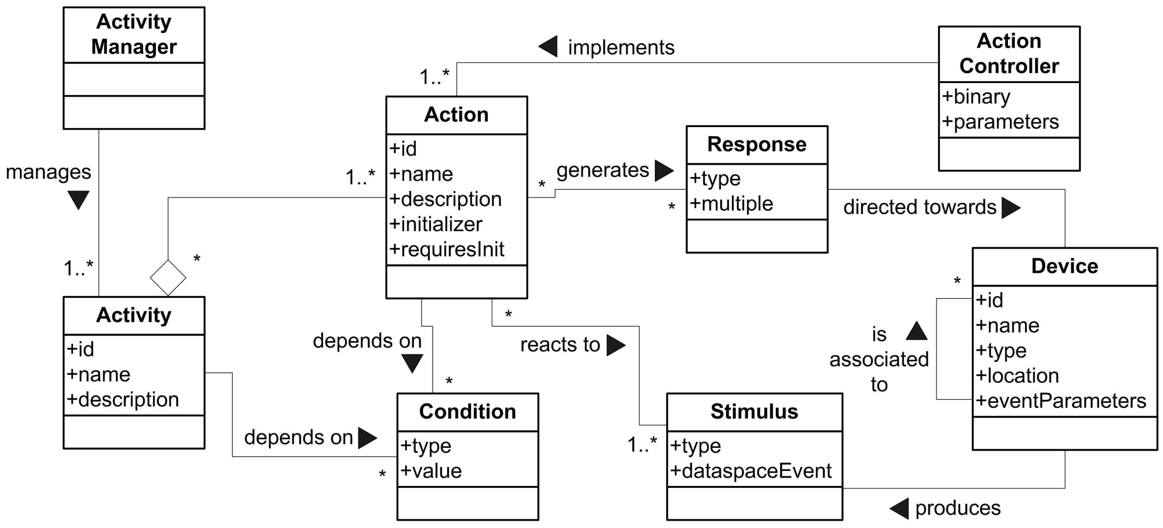


Figure 4.2: Class diagram for the ActivitySpot environment specification

In the environment specification, context is represented only from the perspective of execution conditions, whereas the general architecture also considers the role of context from the point of view of action outcome.

An action may be employed for initializing an activity (*initializer* attribute set to *true*). An action can be set as initializer if it implements all the necessary logic for initializing an activity. Typically, this logic involves processing stimuli that are associated to the execution of initialization steps that may be generic (e.g., selecting an activity) or activity-specific (e.g., populating the visitor profile database with initialization data, such as name, phone number, etc.).

Actions may require or not the activity to be initialized beforehand (*requiresInit* attribute). All the actions that require initialization steps (e.g., feeding the visitor profile with information) or that belong to multiple activities (the action must be framed in the intended activity) should require an activity to be previously initialized. The exceptions to this are, for example, initializer actions, which do not require an activity to be previously initialized because they are actually going to initialize the activity. Another typical scenario for not requiring activity initialization is one in which the place supports a single activity which does not require specific initialization steps. Finally, although the activity model assumes that actions are always part of an activity, some actions may be configured for allowing execution isolated from activity. For example, a place that supports the submission of suggestions by visitors may allow that action to be executed without requiring the visitor to engage into a particular supported activity – the visitor may even be performing an activity that is not anticipated by the space manager.

Stimuli and responses are notified to ActivitySpot components by means of a shared data-space. The data-space supports tuples and several types of events. Interaction or context devices produce and may consume data-space events of a given type and with a given set of parameters. For example, an RFID reading is a simple event having as parameters the reader id and the tag id; a voice response is an event having as single parameter the sentence to reproduce in the sound system.

Stimuli and responses define which stimulus types a particular action is reacting to and, for each particular stimulus type, what response type is going to be generated. Furthermore, a particular response can be generated for multiple recipients (e.g., an RFID stimulus can generate multiple display responses for different displays in the same room).

Please refer to appendix A for an example of each environment specification element.

4.1.2 User interaction architecture

The ActivitySpot user interaction system is based on a device-independent middleware for representing and processing interaction events (see diagram 4.3). Interaction devices are integrated into the ActivitySpot infrastructure by means of device-specific gateways. These gateways implement device-specific protocols for data input/output and bridge communication with the ActivitySpot data-space. Stimulus are converted by the gateway into an ActivitySpot data-space event, while data-space response events are converted to device-specific output data.

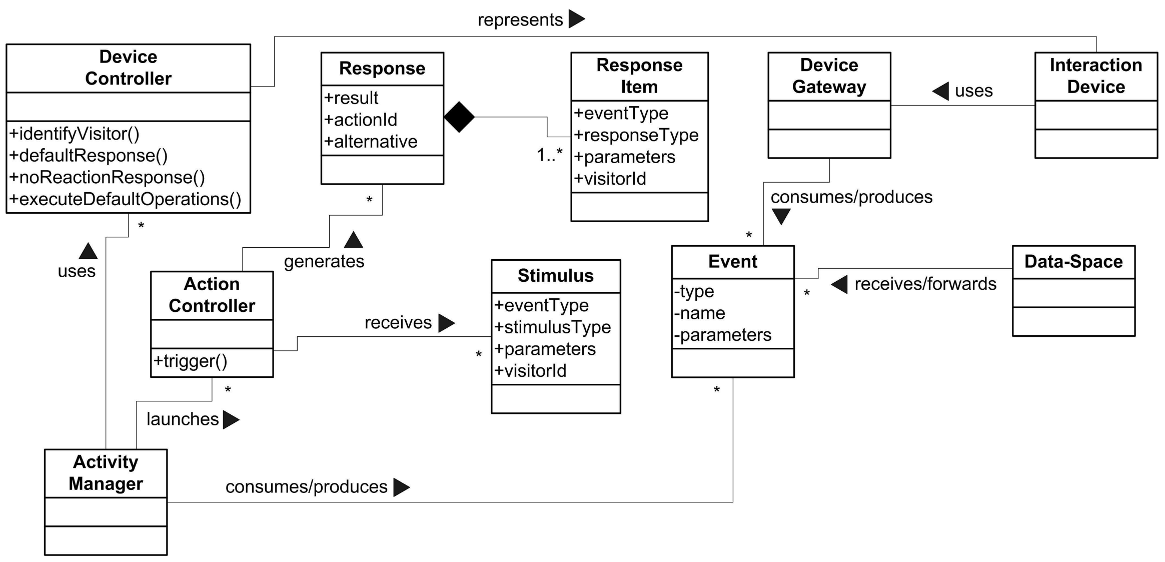


Figure 4.3: Class diagram for the ActivitySpot user interaction system

The Activity Manager listens for interaction events and uses the respective device controller for identifying the visitor. The Activity Manager converts a low-level event into a stimulus and triggers the action controller to which the stimulus is directed. The action controller eventually produces a response, which is sent to the respective output device, through an event at the data-space.

4.1.3 Context architecture

Context-awareness in ActivitySpot comprises an open set of context dimensions. Some context dimensions require a device for capturing context changes (e.g., a temperature sensor) while others can be measured at run-time with system data (e.g., time of day). For the context dimensions that require a sensing device, the architecture is similar to user interaction (see diagram 4.4), with a device gateway generating events at the data-space, which are consumed by the Activity Manager. The Activity Manager then feeds the respective context dimension into the context repository.

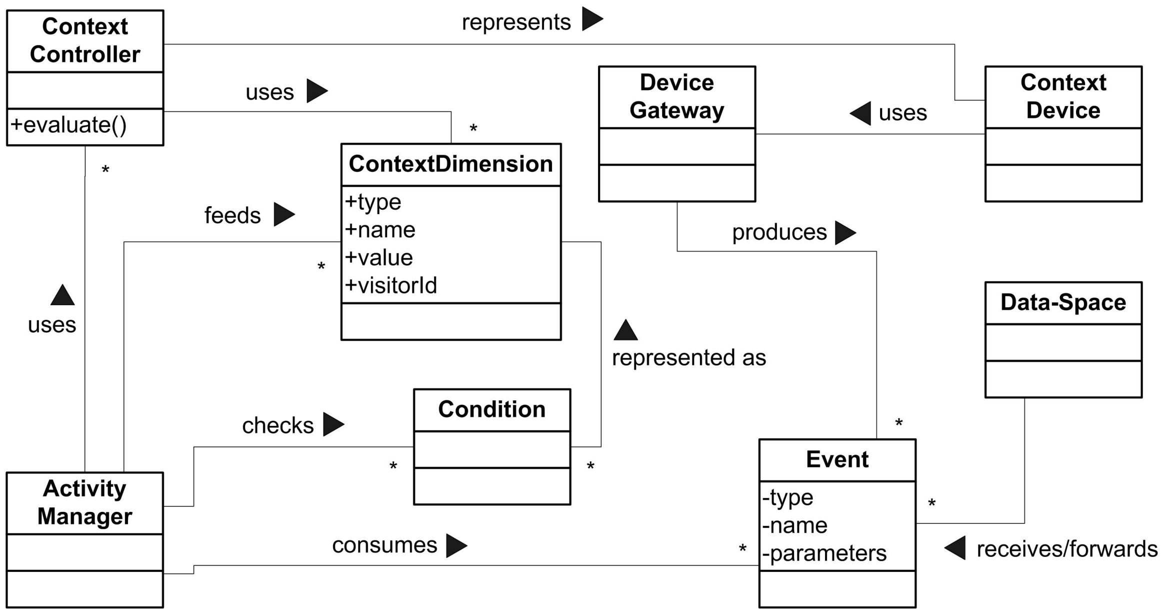


Figure 4.4: Class diagram for the ActivitySpot context-awareness system

Whenever an action or activity execution pre-condition has to be checked, the Activity Manager uses the respective context controller for evaluating whether the condition is met. Context controllers can also be used by the Activity Manager for providing context information during action execution.

4.2 Run-time infrastructure

The ActivitySpot run-time infrastructure provides the run-time mechanisms for managing activity execution supported by ubiquitous computing devices. Its main component is the Activity Manager, which coordinates user interaction and context changes and manages how it contributes to activity execution. Personal information obtained from visitors is also used in this activity management process.

4.2.1 Requirements

The requirements for the ActivitySpot infrastructure can be aggregated into activity management, user interaction, and personalization.

Activity management

The requirements for activity management include issues such as activity structure, activity initialization, activity status, context-awareness, and solution deployment:

1. Simultaneously support multiple activities performed by multiple visitors.
2. Allow action reuse among different activities.
3. Allow both implicit and explicit activity inference, after the first user interaction, providing the necessary activity initialization operations.
4. Keep the status of activity execution, by recording activity history.
5. Allow activity interruption and later resumption.
6. Provide context-awareness at the activity and action levels.
7. The context-awareness model can be easily extended to new context dimensions.

User interaction

The user interaction requirements are related mainly to the expected device heterogeneity and multiplicity of interaction possibilities provided by an ActivitySpot environment:

1. Make no assumption about supported interaction devices types.
2. Facilitate rapid integration of new interaction devices.

3. Hide the low level details of interaction device usage from developers and administrators.

4. Correctly identify a visitor interacting with a device as well as the intended action.

5. Address a response to a stimulus to the correct output device.

6. When multiple output devices are available for a response, select the most appropriate one.

7. Allow for multiple responses for a single stimulus (e.g., complementary responses through different devices).

8. Inform visitors about execution exceptions (e.g., temporarily unavailable actions).

9. Allow the possibility of undoing an action.

Personalization

Personalization requirements are targeted mainly at maintaining visitors' personal information, either to support the ActivitySpot infrastructure (e.g., for user interaction identification) or to influence the outcome of actions:

1. Keep visitors profile, fed at initialization or during activity execution. The profile is kept for further visits.

2. Keep a record of executed actions, in order to profile visitors regarding activity habits (e.g., for making suggestions).

3. Provide support for representing personal context possibly captured by local devices.

It is assumed that visitors are aware of personal information capture and that they can opt for not disclosing access to that information.

4.2.2 Solution

This section shows how the architecture described above is concretized into a run-time infrastructure for supporting activities performed at a particular place. The run-time infrastructure is composed of four nuclear elements: the Activity Manager, interaction devices, the data-space, and action controllers. The details of each element are provided along with the description of the infrastructure behaviour. The following diagram depicts the instantiation of the ActivitySpot run-time infrastructure in a concrete scenario.

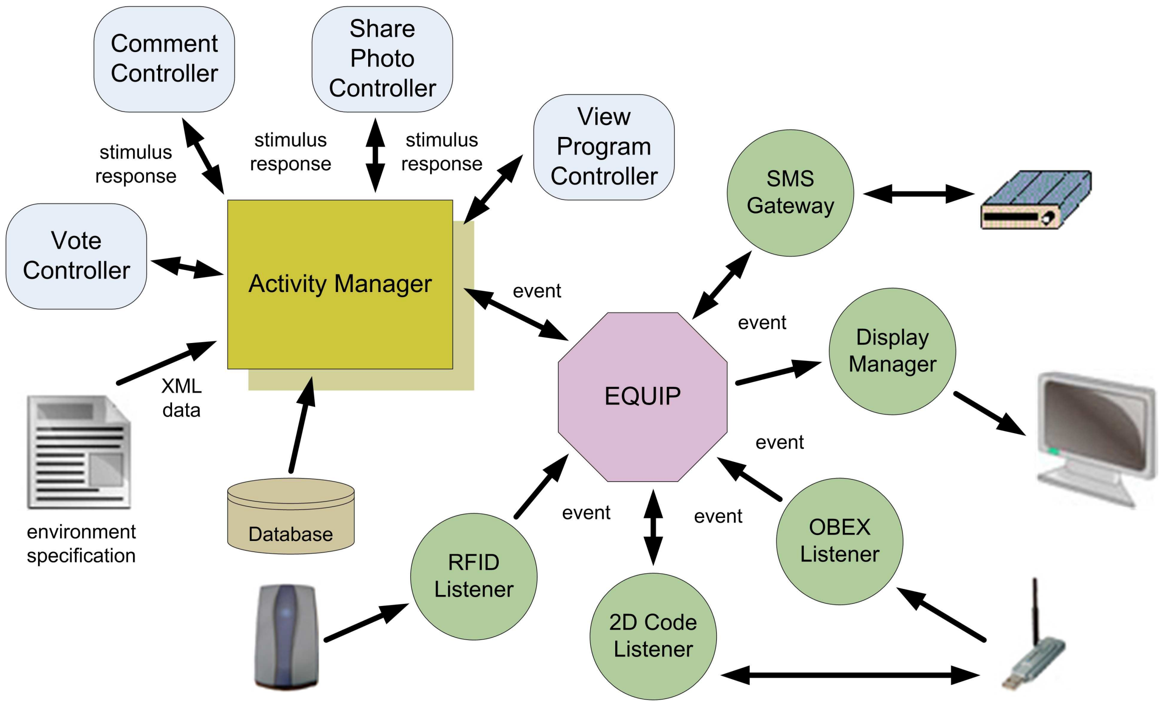


Figure 4.5: An instantiation of the ActivitySpot run-time infrastructure (arrows indicate data flow)

The Activity Manager is a service process that, when started, reads the environment specification (see sub-section 4.1.1) and loads all the information regarding available interaction devices, action controllers, activities, and context dimensions. Action, device, and context controllers are initialized, as well as the connection to the visitor profile database. The environment specification is implemented as a set of XML [W3C 06] documents: one for devices, another for context dimensions, and another for activities and actions specifications[1]. The last step of Activity Manager initialization is registering, at the data-space, for interaction events notification.

The data-space is implemented by an EQUIP data-space developed as part of the EQUATOR project [Greenhalgh 02]. EQUIP was written with reliability and performance as major requirements. At the time of ActivitySpot implementation, EQUIP was fully available only in Java, being determinant in the choice of the language for implementing all the remaining components. EQUIP provides support for tuple-oriented and event-oriented coordination. Tuples and events are both structures defined by a name, a unique id, and a set of data items. EQUIP supports a wide array of data item types, from primitive types (e.g., integer, string, long, byte, etc.) to more complex ones (e.g., date and time, text messages, mouse input, etc.). The difference between tuples and events is that tuples are kept at the data-space until someone removes them whereas events are not persistent. When an event is posted to the data-space, it is immediately disseminated to all the entities that registered for that type of event notification and is removed from the data-space memory. Entities that register for a given event type cannot be notified of past events. On the contrary, when a tuple is added to the data-space, it is kept on the data-space memory, being interested entities notified of that tuple addition and of subsequent operations on it (updates and removal). Entities that register for a given tuple type have access to tuples previously added to the data-space and that were not yet removed.

It is assumed that, for every supported interaction device, there is a corresponding device gateway complying with the user interaction architecture. i.e., generating and consuming interaction events at the data-space. When an interaction device is added to the ActivitySpot infrastructure, a description (name and data items) of the events or tuples it consumes or generates is added to the environment specification. The Activity Manager registers at the data-space for notifications of all the interaction events that are supported by the environment.

Each event or tuple posted to the data-space or consumed by a device gateway has a name and specific data items that describe the interaction details. Whenever a visitor

[1]Please refer to appendix A for an example of each environment specification document.

generates a stimulus through an interaction device, a corresponding stimulus description is sent to the data-space. The Activity Manager senses this stimulus and initiates the stimulus processing, composed of the following steps:

1. loading device controller – based on the event or tuple name, the respective device controller is loaded;

2. stimulus author identification – this is done by invoking the *identifyVisitor* method on the device controller, which uses, for example, a mobile phone number, a MAC address, an RFID code, etc. to perform visitor identification.

3. checking current user activity – this is done by checking the user profile (at a local user profile database) for any ongoing activity; it is assumed that the user had previously initialized an activity, e.g., at registration or by executing an activity initialization action, or that the Activity Manager has previously automatically assigned an activity to that visitor (implicit activity inference).

4. checking activity execution conditions – this is done by evaluating the execution conditions at the activity specification; each condition is evaluated through the *evaluate* method of the respective context controller.

5. (if the current user activity is not set) analyzing all the available activities in order to check whether any can be carried out, i.e., whether it meets its respective execution conditions; an additional condition for automatically assigning an activity to a visitor is obtaining a response from an action as a consequence of that stimulus.

6. (for the current user activity or for each possible activity – see previous step) triggering all the action controllers that react to that type of stimulus and that meet the conditions to be executed, by invoking the *trigger* method – the stimulus description is passed as parameter.

7. collecting results from action controllers, which can be of the following types: no reaction, no response (i.e., the stimulus was processed but no feedback was

produced), and response[2].

8. (in the case no reaction or no response was produced by any controller) executing the default response behaviour defined at the device controller; for example, for an SMS stimulus, sending an SMS stating that the stimulus could not be processed by the system (no reaction) or that the stimulus was processed successfully (no response).

9. (in the case multiple output alternatives were provided in the response description) selecting the most suitable output device – this is done by selecting the first device that is available and that is co-located with the device where the stimulus was made; for example, if an RFID tag carried by the user is intentionally brought near a reader, the user is expecting to see the response in a nearby display, not in a display elsewhere.

10. finally, converting response descriptions into data-space items and posting them to the data-space, which routes the output data to the respective device gateway.

The stimulus reaction behavior is similar to what happens for an event generated by a context sensor. Actions that are sensible to context changes may thus generate a response to an interaction device or, if a response is not suitable, execute some logic without producing any response. Every executed operation (either as a consequence of a stimulus or a context change) is recorded in the Activity Manager user profile and may be later retrieved to check the activity state or to influence the outcome of other operations.

The only requirements of the ActivitySpot infrastructure are a Java Virtual Machine and EQUIP-compliant adapters for each interaction device or context sensor available in the environment. The interaction device and context types supported by the infrastructure are unlimited, given that the infrastructure can be extended (without needing recompilation) to support new types.

[2]though in a well-designed system only a single action controller is going to react to a given stimulus, the infrastructure does not avoid reactions from multiple action controllers

4.2.3 Discussion

The ActivitySpot architecture was designed to meet the requirements of supporting the development and configuration of ubiquitous computing solutions for public spaces. In this section, I discuss how those requirements are met by ActivitySpot.

Activity management

A fundamental research problem of activity-based computing is the inference of the activity the user is willing to accomplish or the user intent. ActivitySpot supports both implicit and explicit activity inference. Implicit inference should however be an option only when the inference error probability is very low. This is true for places where only a few activities can be carried out or where each interaction possibility is associated to a specific activity. At the first interaction, ActivitySpot infers the visitor's activity by analyzing all the possible activities, i.e., all the activities that meet their respective execution conditions, and triggering their action controllers that may react to the stimulus made by the visitor. It is assumed that only a single action controller instance is going to react to the stimulus. The inferred activity is the one to which the action controller belongs to.

Explicit activity inference is achieved after a registration step, which may be achieved: 1) by means of an initialization action, in which the visitor explicitly tells ActivitySpot, through some interaction device, which activity she is engaging in[3]; or 2) at a registration desk, where visitors interact with local staff.

Though explicit activity inference may seem a contradiction with the goal of providing distraction-free, activity-centered support to visitors, it is many times the most appropriate, error-free approach (implicit activity inference is highly prone to inference errors and thus to compromise the overall user experience). The sporadic nature of activities performed by occasional visitors reduces the urgency of automatically inferring activity, because of the low cost of manual, explicit inference. It is a simple task that

[3]ActivitySpot deals with this registration mechanism exactly like it does with implicit inference; only the activity initialization action controller is expected to react to the first stimulus made by a visitor

visitors have to execute, e.g., only at the moment they enter the public space. Nevertheless, explicit activity inference should be based on simple and non-obtrusive mechanisms that do not drive away users at the first contact.

ActivitySpot is able to simultaneously support multiple activities performed by multiple visitors, assuming that the available local interaction devices do not require exclusive, time-consuming usage by a particular visitor. The same interaction device can be used by different visitors for executing distinct actions within distinct activities. The Activity Manager identifies the author of the interaction and triggers the appropriate action for the activity the visitor has registered for. For example, swiping an RFID card over a reader can trigger a "show participant list" for a conference participant and, a few seconds later, a guest of the hotel where the conference takes place can use the same RFID reader for executing a "show restaurant menu" action.

Furthermore, flexibility and reuse in the activity structure, a fundamental characteristic of ActivitySpot, is made possible by the loose coupling between action controllers and the Activity Manager (as well as the environment specification it follows). Different activities may include the same action and thus reference the same action controller (the most typical form of action reuse). But it may also happen that different actions reuse the same action controller. For example, a security inspector is likely to make a recommendation by means of the same operations than a museum visitor would employ to make a comment to an artwork.

ActivitySpot records every interaction made by visitors. Data about stimulus author, type and content, triggered action, response type and content, and date and time of interaction are recorded in an interaction history database. Interaction history serves several purposes: keeping an activity state – some actions may depend on activity state information (e.g., show the restaurant menu only if the visitor has not yet ordered something for lunch) or may be specially targeted at providing activity state; supporting activity interruption and resumption – visitors can later check the state their activity had before interruption; providing activity history – visitors can later remember what they accomplished during their visit; analyzing system usage – public space administrators can mine interaction history data for finding system usage patterns.

Context-awareness is provided both at the activity and action levels. At the activity level, public space administrators can specify execution conditions based on context variables. For example, in an art gallery, people can only visit Bosch paintings from January 24th to March 21st. At the action level, action execution conditions can also be context-based and action controllers can include context-aware logic. The ActivitySpot infrastructure can be easily extended with new context sources, given that they are provided with controllers complying to the ActivitySpot context controller interface.

The extensible character of ActivitySpot – new actions and new context sources can be easily integrated into the framework – and the loose coupling between activities specifications and their logic support is fundamental in enabling evolvable ubiquitous computing systems, which is particularly important in spaces as dynamic as public places, where new social practices are sooner introduced than in other places and where new sensing technologies are sooner put into play. Furthermore, ActivitySpot's agility in face of change facilitates rapid configuration and deployment of new ubiquitous computing solutions for supporting user activities.

User interaction

ActivitySpot is not tied to any particular interaction device. Instead, its user interaction model just assumes that an input device produces stimuli and an output device (which can be the same that produces stimuli) receives responses. This assumption embraces most of the current and future user interaction solutions, because stimulus and response are basic, established user interaction concepts. Furthermore, following the extensible character already described in the previous sub-section for actions and context sources, ActivitySpot facilitates easy and rapid integration of new interaction devices into the local infrastructure. These new devices only have to be provided with EQUIP gateways and controllers enabling the ActivitySpot stimulus-reaction process, including stimulus author identification.

The identification of stimuli authors assumes that visitors, previous to the system usage or during the activity execution, provide the infrastructure with information about their interaction devices, such as providing the mobile phone number through an initialization message or associating RFID tags or a Bluetooth address to a visitor at a registration desk.

The remaining steps of the stimulus-reaction process, described in section 4.2, include: correct identification of the intended action (assuming the environment is configured so that only a single action produces a response to each stimulus); addressing a response to the correct output device (based on the environment specification); selecting the most appropriate device (e.g., the one that is closer or handier) when multiple alternatives are available for a response; and generating multiple responses for a single stimulus (e.g., complementary responses through different devices).

Interaction mistakes or temporary unavailability of a device, for example, are expected. The ActivitySpot infrastructure enables the implementation of action undo/redo, based on user interaction history. Furthermore, if some action is not available (e.g., because it does not have required conditions), ActivitySpot informs visitors about those exceptions. All output device controllers implement generic methods for generating exception responses.

Personalization

Personalization depends heavily on the degree of integration between the personal domain (e.g., visitor profile, personal devices, etc.) and the ActivitySpot infrastructure. To some extent, ActivitySpot is able to provide a personalized experience to visitors, by keeping a visitor profile, fed, e.g., at registration or during activity execution or by keeping a record of executed actions, which enables visitor profiling regarding activity habits (e.g., for later making suggestions to visitors or to adapt action responses). Given the extensibility of the context source infrastructure, ActivitySpot can support the addition of new personal context sources, such as a heartbeat meter, personal agenda, etc.

4.3 Software library

The ActivitySpot software library is targeted at ubiquitous computing developers for the implementation of new actions, either for a specific activity or for any activity for which that action may be executed. The development of the support for a new action involves user interaction – capturing stimuli and generating responses – and action logic – executing operations according to a specific stimulus.

4.3.1 Requirements

The requirements for the ActivitySpot software library include requirements that are common to software libraries in general and requirements that are specific to this work. Software libraries in general should obey to a set of fundamental principles described in [Cwalina 05, Myers 00], namely:

- support to the most common development goals – developers should be able to implement the support for the most common types of action, i.e., interactive actions (with or without a response) and reactive actions (actions that react to some change in the local context, capable to generate an output if appropriated);
- low barrier to entry – developers willing to learn by experiment with the software library should not find obstacles that, to be solved, require a deep knowledge of the library.
- simple development of simple scenarios – the development of simple actions should not require complex initializations;
- self-documenting object models – self-descriptive class, method, attribute and parameter names;
- layered architecture – providing different abstraction layers for different levels of complexity, ideally in different library packages.
- low or no proneness to development errors – the library architecture and development model themselves avoid development errors, by enforcing correct development.

Other requirements are specific to ActivitySpot scenarios:

- extensibility to new interaction/sensing device types – the inclusion of new types of interaction devices in a local environment should not require a new version of the software library.
- abstraction from interaction/sensing device details – developers should not have to know the details of stimulus or context event capture and response submission of a specific device; instead, these details should be abstracted by the library.

4.3.2 Solution

The ActivitySpot architecture described in section 4.1 provides the foundation for the ActivitySpot software library I describe in this sub-section. The software library is exclusively targeted at the development of new action controllers, which implement the behavior of each action available in a place, no matter what activity it belongs to. The implementation of an action controller requires programming the reaction logic to the stimuli and context events to which that action is able to react. The reaction to an event may result in the modification of the system state (e.g., modifying data in a visitor profile) or in the generation of a response directed at the author of the stimulus.

Action controllers must implement a common interface – the *ActionController* interface – defining three methods:

Listing 4.1: Java definition for the ActionController interface

```java
public interface ActionController {
    public void load(String actionId,
                     Properties genericParameters,
                     Properties specificParameters,
                     ActivityManager am);

    public Response trigger(Stimulus stimulus);

    public boolean checkConditions(String visitorId);
}
```

The *load* method implements initialization tasks and is executed when the Activity Manager is initiated and loads into memory all the action controllers for the available actions. The *load* method is invoked with parameters read in the environment specification, namely: the action identifier, generic parameters (common to all instances of the same action controller), and parameters specific to a particular action. An example of a parameter can be, for example, the URL of a web service or the number of items to include in the response. A reference to the Activity Manager is also passed, in the case the action controller requires, for example, access to the visitor profile or to the Activity Manager logging sub-system.

The *trigger* method implements the actual reaction to a stimulus. The method receives a *Stimulus* object describing the stimulus type and content – a stimulus can be an interactive stimulus or a context event – and returns a *Response* object describing the generated response (if any).

The *Stimulus* object is created by the Activity Manager, after sensing an EQUIP event, reading low-level data from it, and converting it into a higher-level representation. Its attributes represent the stimulus type (e.g., "rfid", "sms-in", etc.), the stimulus parameters (e.g., a tag id, an phone number, a message, etc.), and the visitor identification in the local profile database.

Listing 4.2: Java definition for the Stimulus class

```
public class Stimulus {
    private String stimulusType;
    private Hashtable parameters;
    private String visitorId;
}
```

A *Response* object is created and returned by the action controller reacting to a stimulus. This object is then converted by the Activity Manager into a low-level EQUIP representation and posted to the EQUIP server. Its attributes represent the response result (either *NO_REACTION* – the action controller did not react to the stimulus –, *NO_RESPONSE* – the action controller did react to the stimulus, by executing some operation, but did not generate a response –, or *RESPONDED* – the action controller reacted and responded to the stimulus), the action identifier (as defined in the environment specification), an array of response items (different response alternatives or several responses targeted at the same or different visitors), and a flag indicating whether all the response items are to be posted to their respective device (non alternative) or whether only a response item is to be chosen among the alternatives.

Listing 4.3: Java definition for the Response class

```
public class Response {
    public static final int NO_REACTION = -1;
    public static final int NO_RESPONSE = 0;
    public static final int RESPONDED = 1;

    private int responseResult;
    private String actionId;
    private ResponseItem[] items;
    private boolean alternative;
}
```

ResponseItem objects represent the actual response to be posted to an output device. A response type (e.g., “screen”, “speaker”, etc.), parameters (e.g., content URL to display at the screen, content to play at the speaker), and the id of the visitor at who the response is targeted.

Listing 4.4: Java definition for the ResponseItem class

```
public class ResponseItem {
    private String responseType;
    private Hashtable parameters;
    private String visitorId;
}
```

Finally, and optionally, the *checkConditions* method is used to implement any action-specific execution conditions that cannot be specified by means of the standard conditional operators provided by ActivitySpot. For example, an action controller may need to access an external service in order to authorize execution by a particular visitor. This method requires only the visitor id and returns true whether the action can be executed and false otherwise.

Given that a reference to the Activity Manager is passed to the action controller at initialization, developers can access lower-level features in order to implement more sophisticated action logic. For example, they can access the local visitor profile, the EQUIP data-space, and some details of the environment specification, such as supported activities, actions, and devices.

4.3.3 Discussion

The software development tasks required by solutions implemented with ActivitySpot are exclusively centered on the development of the support for new actions. Developers focus on writing action controllers, i.e., writing the logic for reacting to stimuli and, whenever appropriate, producing responses. Though having to think, to some extent, of interaction devices, developers do not have to worry about the low-level details of capturing stimuli from devices or of producing responses.

Developing an action controller requires an effort that depends exclusively on the complexity of the action being implemented. The contract defined by the ActivitySpot library is simple: action controllers have to implement a common interface with only three methods, being one of them optional. The method that deals with the actual interaction has a clear, self-descriptive signature: it receives an object representing a stimulus – either an interaction or a context event – and returns an object representing a response. This simplicity facilitates the development of simple scenarios and does not pose any relevant barrier to developers willing to learn by experiment. Furthermore, the obligation to implement a pre-defined interface and the focus on reacting to stimuli clearly delimitates the scope of development and thus reduces development error proneness.

Advanced developers needing to write more complex action controllers have the means to do so. A reference to the Activity Manager is passed in the *load* method that allows for accessing lower- and higher-level functionality and data. However, at this level of abstraction, proneness to development errors increases.

The extensibility to new interaction/sensing device types is facilitated by the fact that the *ActionController* interface and the *Stimulus* and *Response* classes do not refer any dependency on a particular type of device. Instead, *Stimulus* and *Response* classes contain a string attribute used to identify any device types. The device types supported in a particular place are defined in the environment specification and can be changed anytime.

Finally, as the same action can be part of different activities, the same action controller can be also reused in different activities. Reuse is potentiated, not only within the software developed for a specific public place, but as well at a broader marketplace perspective, for instance, by creating an action controller market, where one could find the support for actions common to many different scenarios.

4.4 GUI authoring tool

The ActivitySpot GUI authoring tool is targeted at public space managers for setting up the support for activities. With the help of a drawing canvas, icons, and a set of forms, this tool enables public space managers to configure: how user interaction and sensing devices are installed throughout the physical space; what actions can be executed and how the available devices enable these actions; and, finally, what activities can be carried out and what actions do compose them. The result of this configuration is an environment specification (see sub-section 4.1.1) that is then used by the ActivitySpot run-time infrastructure for managing user interaction and activity execution.

4.4.1 Requirements

The main requirement for the GUI authoring tool is to facilitate rapid configuration and deployment of new activities and actions. Its interface has to be quickly learnable and effective in helping public space managers to quickly achieve their goals. Most requirements for this tool are similar to those of end-user programming systems [Dey 06, Ko 04], and, although being targeted to a different user population, it also shares with the software library some general requirements:

- low barrier to entry – the tool should be simple enough to allow public space managers to learn by experiment, i.e., it should not intimidate them with an over-complex interface, populated with multiple windows, buttons, or menu options.
- rapid configurations – the tool should facilitate rapid configuration tasks, by providing a simple and clear configuration workflow and hiding from users low level details of devices or actions.
- no proneness to configuration errors – the tool should enforce correct configurations by binding user interaction validations (e.g., form input validation) to the required configuration workflow and configuration scheme.
- extensibility to new interaction/sensing device types – the inclusion of new types of interaction devices in a local environment should not require a new version of the tool.

4.4.2 Solution

The ActivitySpot authoring tool is a Java Swing-based GUI application used for generating environment specifications (see sub-section 4.1.1) for the ActivitySpot run-time infrastructure. The tool is composed of three editors (described in detail in the following sub-sections), each of them corresponding to a step of the configuration workflow (see figure 4.6 for an overview of the tool interface):

- space editor, for drawing the plan of the physical space and positioning interaction or sensing devices in specific locations;
- actions editor, for specifying which actions can be executed and how they are executed, regarding conditions, action-specific parameters, and possible user interaction combinations.
- activities editor, for specifying which activities can be carried out, possible conditions, and what actions compose them.

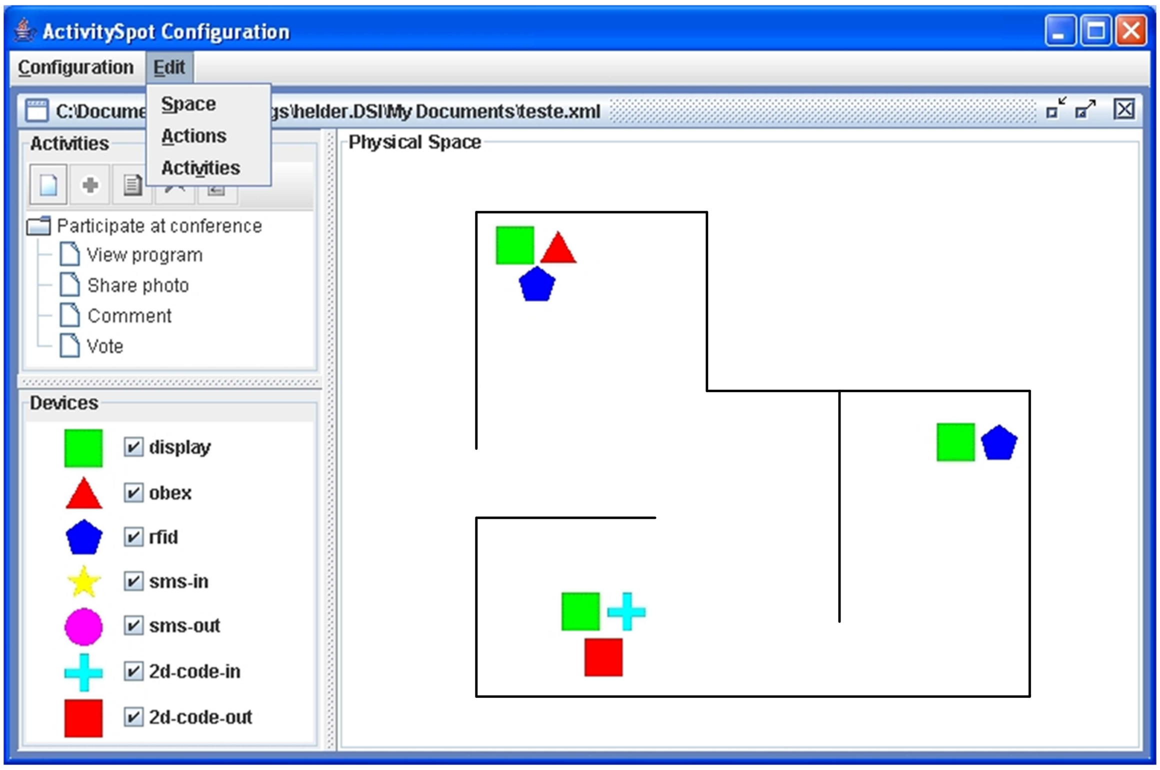


Figure 4.6: An overview of the ActivitySpot authoring tool

The configuration workflow is flexible, i.e., users decide the order of the configuration steps at will and not all the steps have to be executed in each configuration task, except for the first-time configuration, when space, actions, and activities have all to be configured. The space configuration is likely to stabilize soon, because the space and device settings (building plan and devices location) are less prone to changes. The environment specification resulting from a configuration task – in the form of an XML file – is then deployed by the public space manager to the ActivitySpot run-time infrastructure. This deployment process is manual (copying the file to the ActivitySpot run-time base directory), though an ideal solution would be a transparent action executed in the tool (for example, an "update specification" menu or button that would transfer the specification to the run-time environment).

For the sake of extensibility, some resources needed for the authoring process – available action controllers and supported device types and context dimensions – are only attached to the tool at run-time, after processing the authoring tool configuration. This approach is similar to the one adopted in the iCAP authoring environment [Dey 06].

Space editor

The space editor (see figure 4.7) creates a representation of space that reflects the distribution of interaction and sensing devices throughout the physical space. This editor has two panels – devices and plan. The devices panel lists all the device types supported by the physical space[4]. This information is obtained from an XML configuration document that details all the supported devices: input/output, interaction/sensing event parameters, and a run-time reference to the device controller. This document is manipulated each time the infrastructure is extended with a new device type. Nevertheless, this low-level configuration is transparent to the authoring tool user; only the device type is visible, as well as an associated colored shape that eases device recognition in the plan.

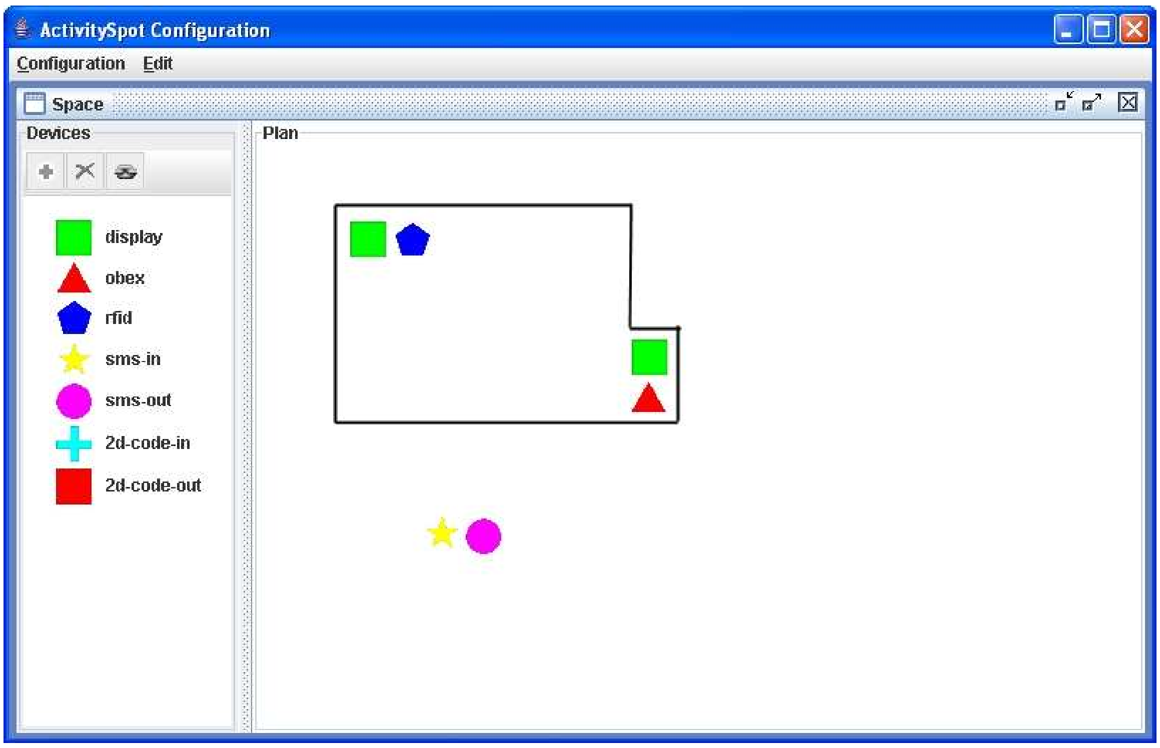


Figure 4.7: The space editor view of the ActivitySpot authoring tool

The plan panel is used to represent the physical plan of the building or outdoor area. It is implemented as a free-drawing canvas, based on the SATIN [Hong 00] toolkit, where users can employ basic sketching techniques for representing the physical space:

[4]This does not necessarily correspond to the actual number of devices, because, for each device type, zero or multiple devices may have been installed in the physical environment.

dragging the mouse over the canvas for drawing a line, clicking over a line for selecting it, dragging the mouse over a selected line for moving it, or clicking over a selected line and pressing the delete key for erasing it.

Devices are added to the plan panel by double-clicking on it or by selecting it and then clicking on the add button on the toolbar. Devices can be moved in the plan panel just by dragging it and can also be deleted by selecting it and clicking on the delete button.

The tool allows for the specification of device associations. An association between devices establishes a strong interaction workflow, typically between input and output devices. For example, associating a particular RFID reader to a display informs the ActivitySpot infrastructure that, whenever a stimulus is detected for that particular RFID reader and the resulting action produces a display response, this response has to be routed to that particular display. Associations are created by selecting the respective devices in the plan panel and then by clicking on the link button. Associations can as well be deleted. Figure 4.8 represents an example of a device association.

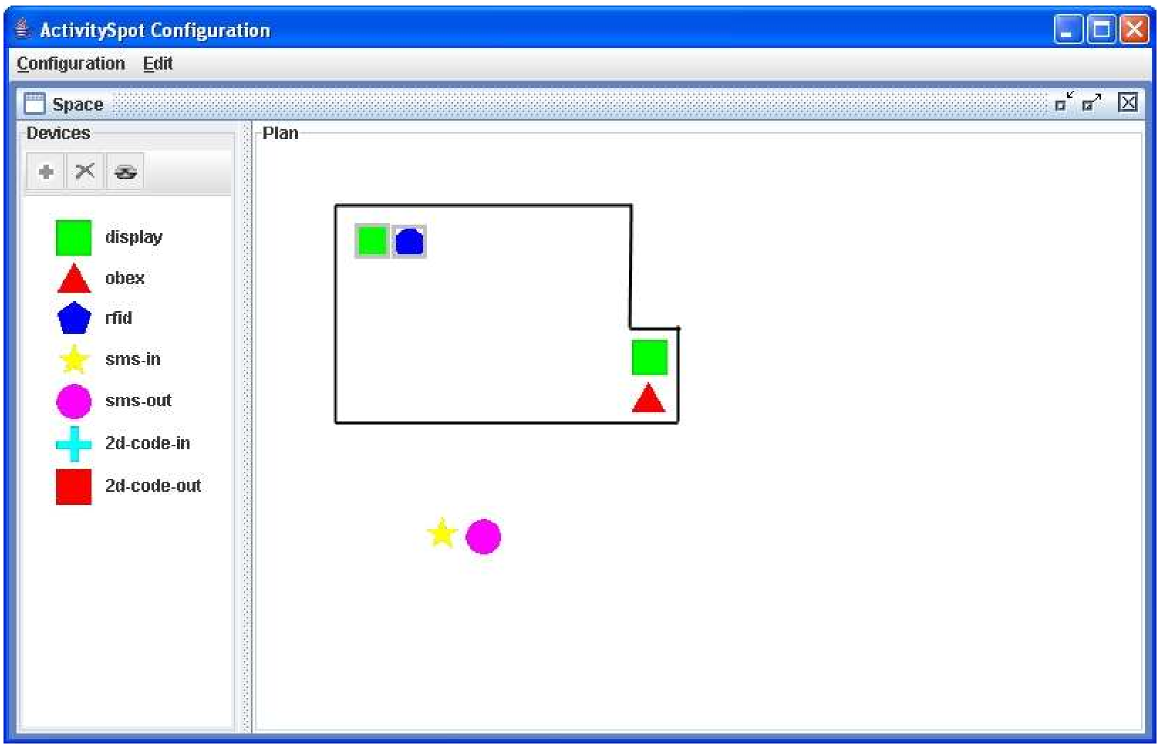
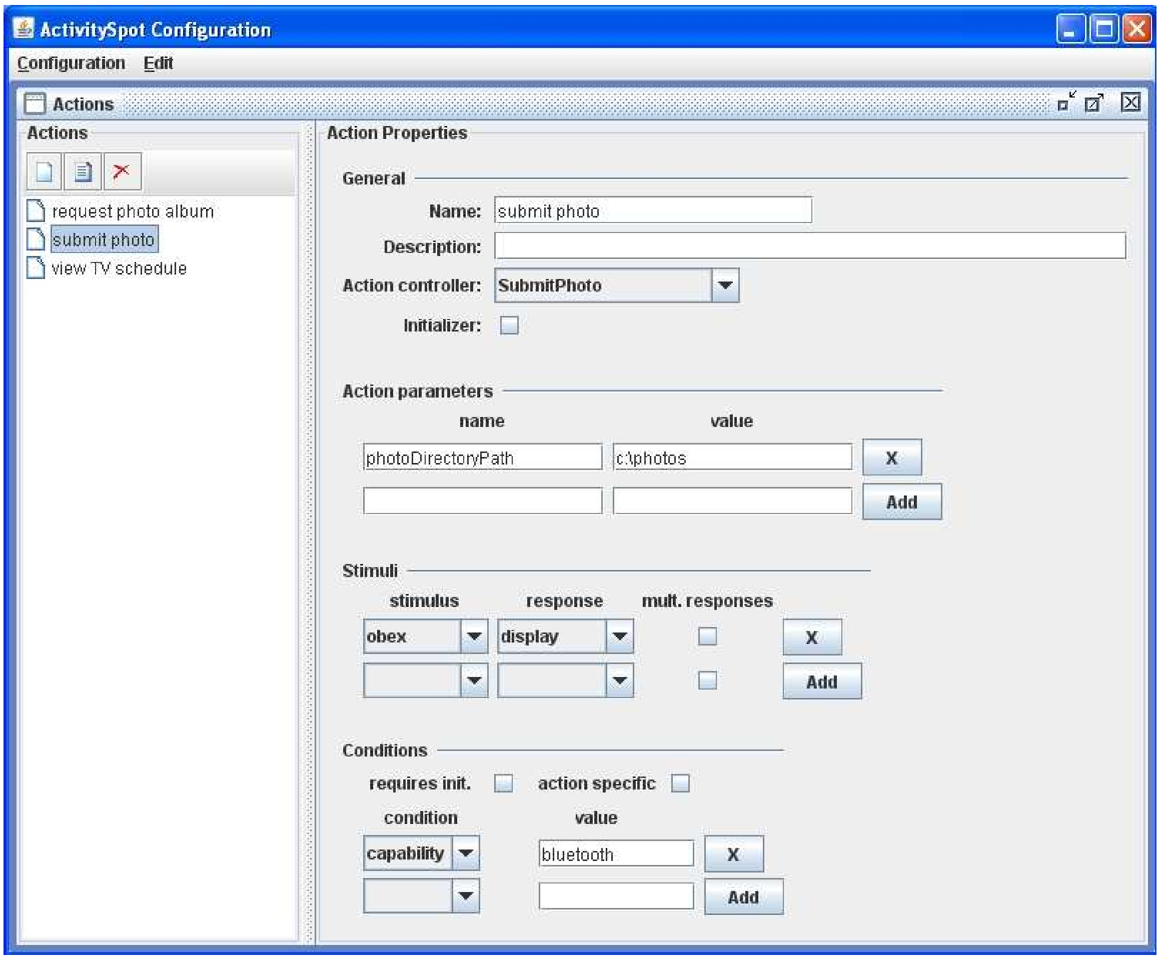


Figure 4.8: An example of device association

Actions editor

The actions editor allows for the specification of the actions that an ActivitySpot-enabled place supports. The actions editor has an actions panel listing the actions that the user has specified for a particular configuration. This panel has three buttons (see figure 4.9), allowing respectively for creating a new action, copying and deleting an existing action.



Actions: request photo album; submit photo; view TV schedule

General
Name: submit photo
Description:
Action controller: SubmitPhoto
Initializer: ☐

Action parameters

name	value
photoDirectoryPath	c:\photos

Stimuli

stimulus	response	mult. responses
obex	display	☐

Conditions
requires init. ☐ action specific ☐

condition	value
capability	bluetooth

Figure 4.9: The actions editor view of the ActivitySpot authoring tool

When a particular action is selected, the action properties panel is activated. This panel is a form where the user specifies general properties, action parameters, supported stimulus-response pairs, and action conditions[5]. Some notes on the action properties editor are nonetheless worth of mention:

[5]Refer to sub-section 4.1.1 for details on action properties.

- In the general properties section, the action controller class is selected among a library of previously installed action controllers. The action controller library can be extended anytime.

- Action-specific parameters (such as a file system path for a photo repository), which are passed at run-time to the action controller, can be specified by adding name-value pairs.

- In the stimuli section, where users add stimulus-response pairs, several entries can be made for the same stimulus type provided that the response types are different.

- The action-specific marker, in the conditions section, means that the ActivitySpot infrastructure has to invoke (or not) the *checkConditions* operation onto the action controller in order to evaluate whether the environment fulfills action-specific conditions (e.g., the availability of a particular web service).

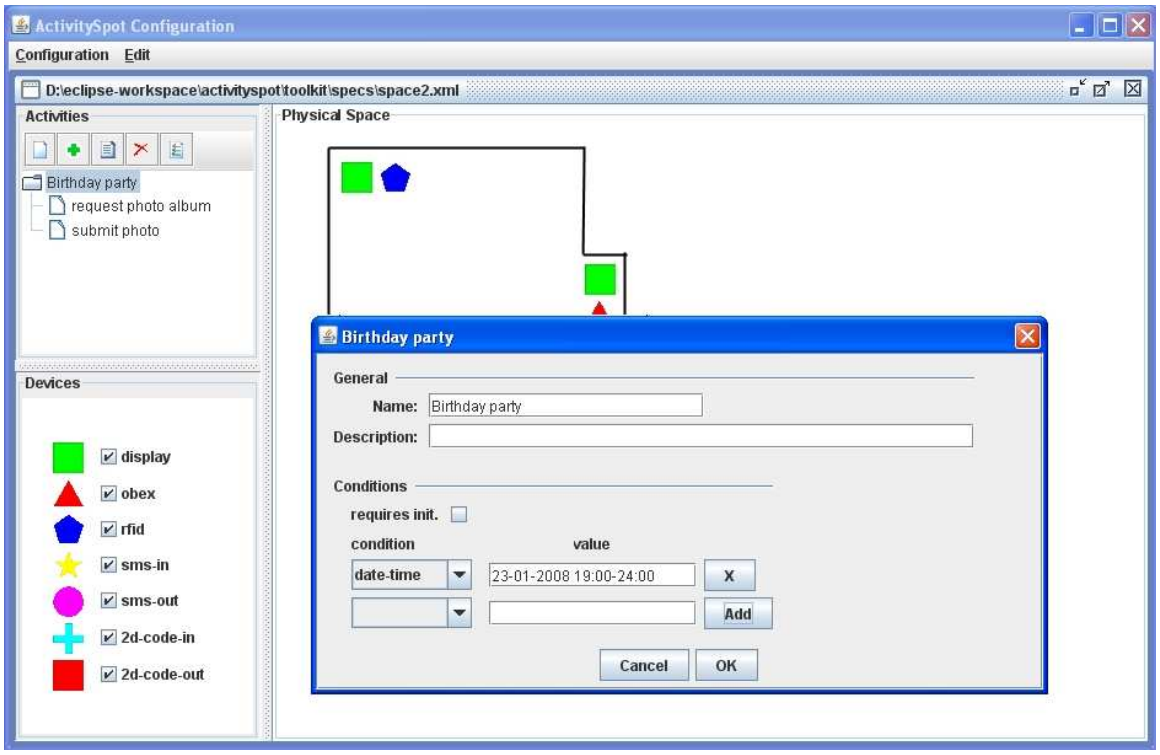
- Context-dependent conditions (added by the user in the conditions section) are based on the context dimensions that the infrastructure is able to support. The only context dimension that is native to the infrastructure is the date/time one. All other dimensions are extensions to the infrastructure and depend on the provision of context controllers and, in many cases, sensing devices providing context data. Therefore, the condition combo box is loaded at run-time with the supported context dimensions (read from an XML configuration document). In the example shown in figure 4.9, a capability condition is set, meaning that the "submit photo" action can only be executed if the visitor profile indicates that some personal device is Bluetooth-enabled. When more than one context-dependent condition is set, the default logical operator is *and*.

Activities editor

The activities editor allows for the specification of the activities that a place supports. Activity specification comprises generic properties, the actions composing it, and execution conditions. The activities editor is composed of an activities panel, a devices panel, and a physical space panel. The physical space panel is just a frozen visualization of the plan drawing and device locations specified in the space editor.

The devices panel is again based on the device types that are supported by the infrastructure, but is here used just as a legend for the physical space panel. The check boxes associated to each device type are just device visualization selectors – when enabled, that specific device type is visible in the physical space panel; when disabled, every instance of that device type is hidden. By using device selectors, the space manager can evaluate more clearly how and where a particular device type is going to be used, disabling the visualization of all other devices.

The activities panel has five buttons allowing for generic operations on activities or on its actions, respectively: create a new activity; add an action to a selected activity; copy a selected activity; remove a selected activity or action; and edit the properties of a selected activity or action. When adding an action to an activity, the user is presented with a combox box containing all the actions specified in the actions editor, except those that have already been added to that activity.



D:\eclipse-workspace\activityspot\toolkit\specs\space2.xml

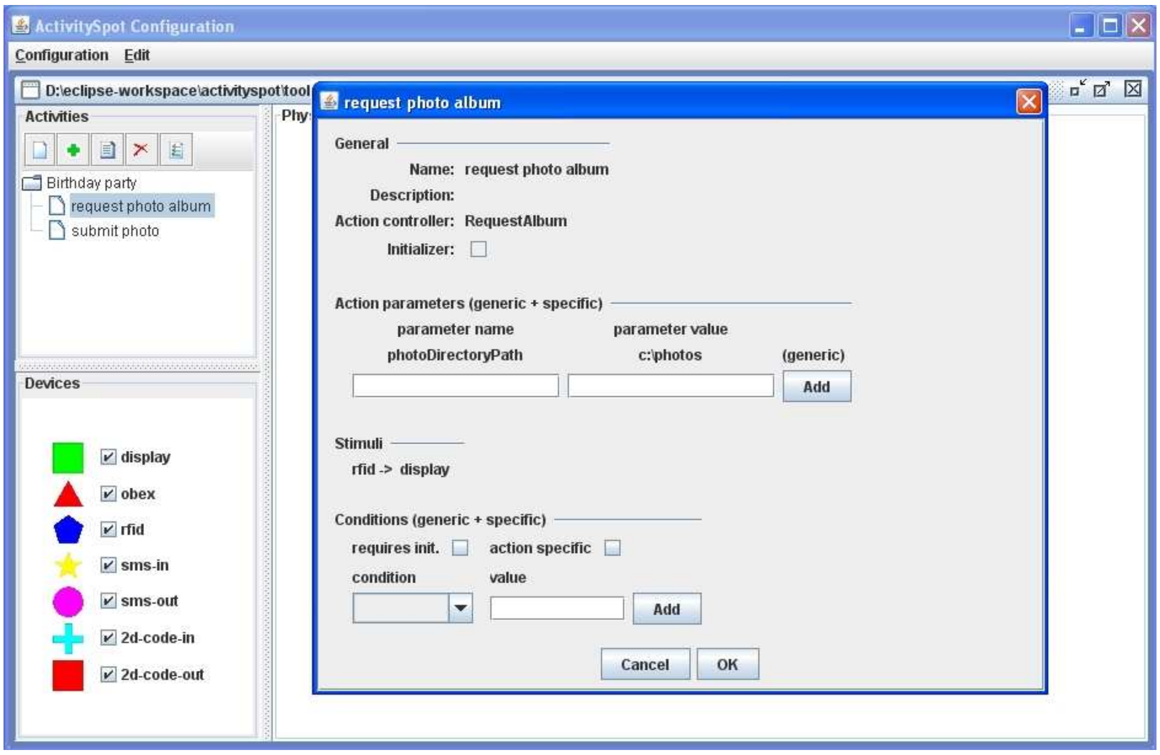
Birthday party
- request photo album
- submit photo

Name: Birthday party
Description:
requires init. ☐
date-time: 23-01-2008 19:00-24:00

Figure 4.10: The ActivitySpot activities editor and the activity properties box

The activity properties box (see figure 4.10) allows for the specification of general properties – name and description – and execution conditions. An activity may require to be initialized before being carried out (e.g., it may need some initialization procedure to be accomplished before actual actions are executed). As detailed in the previous sub-section for actions, activity execution may also be context-dependent.

The action properties box (see figure 4.11) allows for the specification of action properties that are specific to the activity in which the action is executed, i.e., these properties have no effect if the action is employed in another activity. All properties are inherited from the generic specification made in the actions editor. Some properties, such as the action controller or supported stimuli and responses, cannot be specialized, because they are strongly dependent on the action controller implementation, which is common to all instances of that action. However, action parameters and execution conditions support further specialization.



General

Name: request photo album
Description:
Action controller: RequestAlbum
Initializer: ☐

Action parameters (generic + specific)

parameter name	parameter value	
photoDirectoryPath	c:\photos	(generic)

Stimuli

rfid -> display

Conditions (generic + specific)

requires init. ☐ action specific ☐

condition value

Figure 4.11: The ActivitySpot activities editor and the action properties box

4.4.3 Discussion

The ActivitySpot authoring tool provides public space administrators with a high-level, simple, and comprehensive interface for specifying how an environment instrumented with ubiquitous computing devices supports its visitors. Its activity-centered character, common to the whole ActivitySpot framework, keeps users focused on the activities enabled by the infrastructure. They only manipulate higher level concepts and, though working with device representations, they do not have to know their technical details. Moreover, the tool concepts are coherent with those employed in other components of the framework, facilitating communication between public space administrators and action controller providers.

The simplicity of the user interface facilitates learning and exploration by users with little computer expertise. Furthermore, it makes rapid configurations possible with only a few steps. However, this simplicity necessarily comes along with limitations: tool users compose available resources (devices, action controllers, and contextualization mechanisms) but have little influence on the response of the environment to visitor interactions. This power is almost completely on the action controllers' implementation.

The ActivitySpot authoring tool can be easily extended with new devices, action controllers, or context dimensions. All that is required is to edit configuration documents (for devices and context dimensions) or to copy/download action controllers to a specific directory. Extensibility is nowadays a major feature: new technologies come in to the market at high pace and need to be integrated with existing infrastructures. Moreover, public spaces and buildings are evolvable and thus require management tools that adapt to changes in the physical environment and in the activities there supported [Rodden 03, Tolmie 02].

The tool has clear user interface limitations, regarding the general look and feel and some details in functionality (e.g., more logic operators and combinations for context-dependent conditions and validation of action parameters and conditions values). Although the simplicity of its interface contributes to reduce the tool's proneness to usage errors, more could be done for assuring error-free environment specifications (e.g., enforcing a complete specification of the three dimensions – space, actions, and activities

– or alerting the user for interaction devices that are not employed in the specified actions). General user interface concerns were certainly not the focus of this work, but would surely be a matter of extreme importance in case this tool is released to public usage.

4.5 Summary

The conceptual model described in chapter 3 was realized into the ActivitySpot software framework for supporting activity-based ubiquitous computing solutions in public spaces. ActivitySpot is composed of a run-time infrastructure for managing user interaction and activity execution, along with a software library for developing the support to new actions, and a GUI-based authoring tool, targeted at public space administrators, for specifying which devices, actions, and activities the space is providing to visitors.

This chapter described those three components of the framework, eliciting for each one its requirements, providing all the details of the proposed solution, and concluding with a discussion of the solution.

Chapter 5

Evaluation

The ultimate goal of this work is to validate the thesis stated in section 1.2.3. The validation of my thesis involves evaluating how far the proposed model of activity and user interaction: a) can be implemented in a ubiquitous computing infrastructure; b) enhances the visitor experience, making the visitor feel effectively assisted in the activity in hands, without distracting him or her; c) provides public space administrators with productive and manageable means for deploying or reconfiguring the ubiquitous computing support to human activities; and d) eases the task of ubiquitous computing specialists in developing the support for new assistance features.

The evaluation strategy addresses these four evaluation goals in three different stages:

1. implementation of the ActivitySpot run-time infrastructure – this stage demonstrates that the proposed model of activity and user interaction is implementable in an effective ubiquitous computing infrastructure.

2. evaluating user experience in ActivitySpot-enabled environments, by conducting several end-user studies, based on real ubiquitous computing scenarios – this stage demonstrates that the ActivitySpot framework is able to enhance the visitor experience without additional, dispensable distractions, and that little effort is required for the development of the support for new activities and actions.

3. evaluating ActivitySpot environment management with the GUI authoring tool, by conducting a user study with several subjects – this stage demonstrates that

the proposed model is fundamental in facilitating the task of public space administrators in deploying and managing a ubiquitous computing infrastructure and supported activities.

The details and the discussion of the implementation of the ActivitySpot run-time infrastructure, described in section 4.2, by itself characterize the first evaluation stage. The following two sections detail and discuss the remaining evaluation stages. A complete repository of surveys, reports, results spreadsheets, and materials (flyers, posters, instructions, etc.) that supported each evaluation stage can be found on-line [Pinto 08].

5.1 User experience in ActivitySpot-enabled environments

The evaluation of the end-user experience with ActivitySpot-enabled environments takes into account whether the conceptual model and the user interaction I propose is adequate to the cognitive challenges faced by occasional visitors to public spaces. The occasional nature of this interaction – many visitors are probably going to interact with that ubiquitous computing environment only once – implies a very short learning time. Moreover, visitors should not be distracted from the activity that brought them to that particular place. Preferably, the ubiquitous computing environment should provide visitors with an interaction model that requires little cognitive effort. The reasonability of this effort strongly influences how well the ubiquitous computing environment integrates with the visitor activity and the time required for learning how to use the system.

An additional aspect that must be taken into account in this evaluation is the amount of personalization perceived by visitors, how much it meets their expectations, and the effort visitors are willing to commit for a higher personalization.

With these high-level concerns in mind, and based on the user interaction and activity modelling challenges described earlier (see section 1.3) and on several reference evaluation models [Compeau 95, Kaptelinin 99, Venkatesh 03, Scholtz 04], the following evaluation goals were defined (each evaluation goal is met by a set of premises):

- compatibility of the conceptual model:

  - (C1) visitors understand the assistance that is being offered to their activity, i.e., they understand what the ubiquitous computing environment is providing them and how it can support their activity;
  - (C2) visitors find that the conceptual model of the provided assistance is compatible with their own mental model of the same activity;
  - (C3) visitors understand that all interactions are integrated into their activity, i.e., every interaction with the ubiquitous computing environment is understood as being part of the interaction flow of their activity.

- user interaction:
  - (I1) visitors are able to successfully execute actions without any previous training or help other than the concise visual instructions provided to them;
  - (I2) visitors consider that the system responds to their stimuli in a timely and predictable manner;
  - (I3) visitors consider that the effort required by the system does not divert them from their activity;
  - (I4) visitors consider that the steps required for initializing their activity is not disruptive.

- usefulness:
  - (U1) visitors consider that the system helps them achieving the goals for their activity, preferably more effectively when compared to alternative situations (conventional assistance, single application in mobile phone, and interactive kiosk);
  - (U2) visitors consider that the personalization provided by the system is adequate to their needs.

ActivitySpot was evaluated in three different user studies, collecting data from surveys, observation, and log analysis. The evaluation method (sampling, survey items, and collected logs) gradually evolved along the three user studies, either as a consequence of the diversity of the evaluation scenarios or just with minor improvements in order to better fit the evaluation goals. I further describe each of the user studies and conclude the section by discussing the evaluation results.

5.1.1 PhD poster session

The first user study was conducted during a one-day PhD poster session integrated in the Annual Engineering Week (October 2005) at the University of Minho. Two different activities were supported by ActivitySpot: visiting the poster session and presenting a poster. Both activities took place in the poster exhibition area. Although in both cases many users were university members or students, the scenario, as an extraordinary event, provoked the situation that characterizes this work: novelty of activity, physical setting, and infrastructure support.

ActivitySpot was evaluated by 15 users (4 women and 11 men), with ages ranging between 24 and 44. As this work explores how people deal with a system that they not anticipated, I limited awareness of the system itself before the evaluation took place. This compromised in advance user sampling, because subjects would inevitably anticipate the system. Therefore, user sampling was not controlled, waiting for evaluation subjects to naturally use ActivitySpot, as it happens in a real setting. Leaflets describing ActivitySpot and the supported activities were left all over the poster exhibition area and people just dropped by the registration desk to volunteer. At registration, participants were given an evaluation survey (see appendix B.1 for details on the survey design) to be returned at the end of the poster session.

Interaction means and activity initialization

The proposed interaction means explored participants' mobile phone capabilities (SMS, Bluetooth, and infra-red connectivity), public displays, and RFID. An SMS gateway (based on jSMSEngine [SMSLib 08]) was listening to a GSM modem for SMS input and used the same modem for SMS responses to visitors. Within the exhibition area, two interaction spots were available, each with a computer running a public display (overhead-projecting to the walls) and equipped with an RFID reader and Bluetooth and infra-red sensors. The content presented in the display was run by a Situated Portal [José 04], a platform for context-ware public displays. Close to interaction spots, small posters advertised how to use the available sensing technologies to interact with ActivitySpot. Each participant was lent two RFID tags – keyring-like and credit-card-like.

Participants who enrolled in the user study had to explicitly choose their activity by sending an initialization SMS message to the ActivitySpot SMS center (see next sub-section). Then, they went to the registration desk in order to obtain their pair of RFID tags that later allowed them to execute particular actions. Each RFID tag was explicitly associated by the registration staff to the mobile phone number used in the initialization step.

Proposed actions

The same leaflets used for inviting people to use ActivitySpot provided short instructions about the proposed activities and actions:

- initializing activity – participants send an SMS with the syntax *init <vst—apt>* (e.g., *init vst* for visiting the poster session and *init apt* for presenting a poster). The same stimulus registers the participant into the system and initializes the activity. The response to this stimulus is a welcoming message and a password for authenticating in the ActivitySpot web site.

- having an overview of the exhibition (for both activities) – at an interaction spot, participants swipe their RFID card over the reader and see the response in the wall-projected display. The response content includes an exhibition plan, a listing of the three most popular posters (a combination of the most voted and most bookmarked), a picture randomly selected among those shared by the participants, and activity-specific information: the next event in the Engineering Week program (for poster visitors) or a request for voting (for poster presenters who did not yet vote).

- commenting a poster (for both activities) – participants send an SMS with the syntax *cmt <poster id> <comment>*. Each poster has a unique identifier. The response to this action is a confirmation message[1].

- voting for a poster (only for poster presenters) – participants send an SMS with the syntax *vot <poster id>*. Each participant is allowed to vote only once. The response to this action is a confirmation message.

- sharing a photograph (for both activities) – participants owning a camera- and Bluetooth- or infra-red-enabled mobile phone send a photograph to the Bluetooth or infra-red sensors located at an interaction spot. The response to this action – a confirmation message along with the shared photograph – is presented in the wall-projected display.

- bookmarking a poster (for both activities) – participants send an SMS with the syntax *mrc <poster id>*. The response to this action is a confirmation message.

- viewing my run (only for poster visitors) – at an interaction spot, participants swipe their RFID keyring over the reader and see the response in the wall-projected display. The response content includes a listing of bookmarked posters, comments made to posters, and a reference to posters sharing keywords with the posters bookmarked or voted by the visitor.

- viewing my poster (only for poster presenters) – at an interaction spot, participants swipe their RFID keyring over the reader and see the response in the wall-projected display. The response content includes the comments to the poster made by other participants, the number of votes, and how many times the poster was bookmarked.

Participants could later log into the ActivitySpot web site and view the same content that could be seen in the wall-projected displays as well as details of bookmarked posters (author e-mail and a link to the poster file). The web interface could be very useful particularly for poster presenters who could feel embarrassed to see the reaction to their poster in the wall-projected displays.

[1]ActivitySpot automatically responds with a warning message if participants mistype their SMS

5.1.2 Cultural center

In this scenario, a six week long study held in the first trimester of 2006, at the Vila Flor Cultural Center, in Guimarães, ActivitySpot was deployed to assist spectators at three different moments of the shows: before, at the interval, and afterwards. ActivitySpot was run in a total of 19 shows taking place at two different theaters in the cultural center. A single activity was proposed, composed of actions allowing spectators to obtain detailed information or give feedback about the current show.

During the period ActivitySpot ran in the Cultural Center, a total of 24 participants (18 men and 6 women), with ages ranging between 21 and 39, volunteered for participating in the study. In order to engage participants, their effort was compensated with tickets for shows. Most visitors spontaneously addressed themselves to the ActivitySpot registration desk after reading leaflets or looking at public displays' advertisements. Some other participants knew ActivitySpot by other means (local press, Vila Flor Cultural Center web site, etc.) and pre-registered at the ActivitySpot web site, where they could provide, besides general personal data, information about their entertainment preferences.

Participants could choose between using ActivitySpot only once or as many times as they attended shows in the Cultural Center. In the latter case, registration to ActivitySpot was made only once. At registration, participants were given a survey (see appendix B.2 for details on the survey design) that they returned after the last show they attended to.

Interaction means and activity initialization

Besides the SMS gateway already used in the previous study, three interaction spots were installed at the entrance hall of two theaters (two interaction spots in the Grande Auditório theater and another in the Pequeno Auditório one). Each interaction spot was equipped with a computer running a public display (using the LCD wide screens available in the entrance halls) and connected to an RFID reader and a Bluetooth sensor. A 2D-code reader application (based on the TRIP project [de Ipiña 02]) was provided on request to visitors owning a Bluetooth- and camera-equipped, Java-compliant mobile phone. This application was used to capture 2D-codes stuck to the entrance hall walls and pillars and to send the code over Bluetooth to a 2D code gateway. Each 2D code was associated to a specific action.

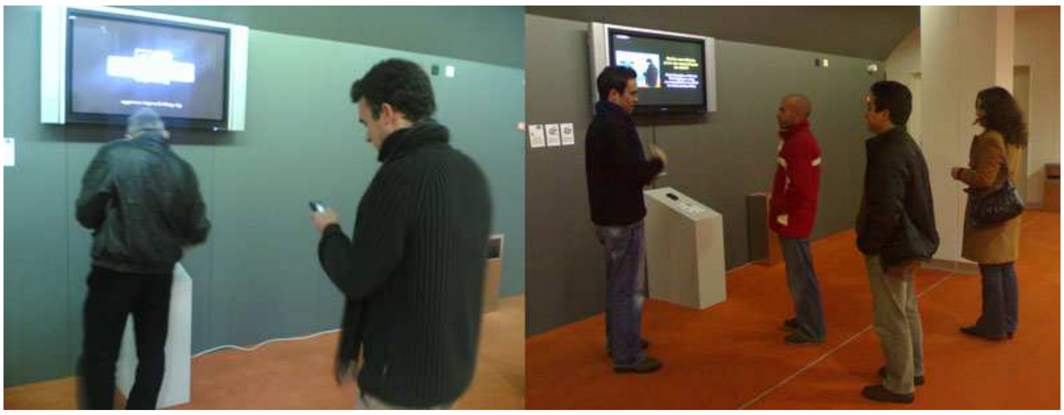


Figure 5.1: ActivitySpot being used at the Cultural Center

At registration, visitors were asked to provide their name, mobile phone number, and were given a pair of RFID tags (keyring-like and credit-card-like) and a leaflet describing what actions were available. Short instructions about system usage were spread near the interaction spots. Visitors owning a compliant mobile phone could also receive (through Bluetooth push) the 2D-code reader application and install it in their mobile phone. This process made possible the association of a visitor identity to the respective Bluetooth MAC address.

Since there were no simultaneous shows and, consequently, only a single activity was supported at each time, ActivitySpot implicitly inferred the intended activity, i.e., the activity was automatically initialized for the current show after the first interaction made by the visitor.

Proposed actions

Visitors had different interaction alternatives for executing the actions composing their activity:

- viewing details about the current show – at an interaction spot, participants swipe their RFID card over the reader and see the response in the wide screen above.

The response content includes general information about the show (cast, credits, duration, a picture, etc.) and information resulting from visitor interaction (comments made by visitors and vote average).

- commenting the current show – participants send an SMS with the syntax *mensagem <comment>*. The response to this action is a confirmation message.

- voting for the current show – participants can either send an SMS with the syntax *vota <a value between 1 and 5>* or capture a 2D-code corresponding to the intended vote. Each participant is allowed to vote only once. The response to this action is a confirmation message (either through SMS or in the 2D-code reader application interface).

- sharing a photograph – participants owning a camera- and Bluetooth-enabled mobile phone send a photograph to the Bluetooth sensor located at an interaction spot. The response to this action – a confirmation message along with the shared photograph – is presented in the wide screen above.

- viewing my show – at an interaction spot, participants swipe their RFID keyring over the reader and see the response in the wide screen above. The response content includes comments made the participant herself, suggestions about other actions (based on the personal interaction history for that specific activity), a list of other participants attending to the show that share preferences with the participant, and the next scheduled show of interest (again based on personal preferences).

- viewing details about an upcoming show – at an interaction spot, participants capture a 2D-code corresponding to the intended upcoming show and see the response in the wide screen above. The response content includes general information about the show (cast, credits, duration, a picture, etc.).

Participants could later log into the ActivitySpot web site and view contributions made during the shows, such as comments and vote average for each show, shared photographs, and the top ten rated shows. They could also view their own show attendance history.

5.1.3 Conference

The last user study was held during a three day conference on human-computer interaction (October 2006). Three different activities were supported, depending on the goals of conference participants: authors presenting their work, conference organizers, and conference participants who were not presenting any work (as main authors).

Some improvements were introduced in this study, based on lessons learned in the two previous studies. The major goal of these improvements was to achieve a better interaction design, looking for understanding visitor needs. Prior to the study itself, I made an activity analysis, by submitting surveys to people who usually participate in conferences, in order to obtain their view of the activity, i.e., which goals they establish and which actions they execute in order to accomplish those goals. This information helped me identifying the actions that could better meet user needs. After this phase, a prototype description (interaction details for each available action) was evaluated by a human-computer interaction expert, who identified some minor interaction problems.

A total of 8 participants (7 men and 1 woman), aging between 25 and 42, used the system and answered the surveys (see appendix B.3 for details on the surveys design). A second group of participants (6 people) was selected as the control group, in order to assess how the ActivitySpot assistance contributed to achieve activity goals, compared to the conventional assistance available in conferences.

Conference participant data was obtained beforehand in order to build a basic profile (name, institution, and work authorship) that was used as a source for the content of some actions responses and for speeding up visitor registration.

Interaction means and activity initialization

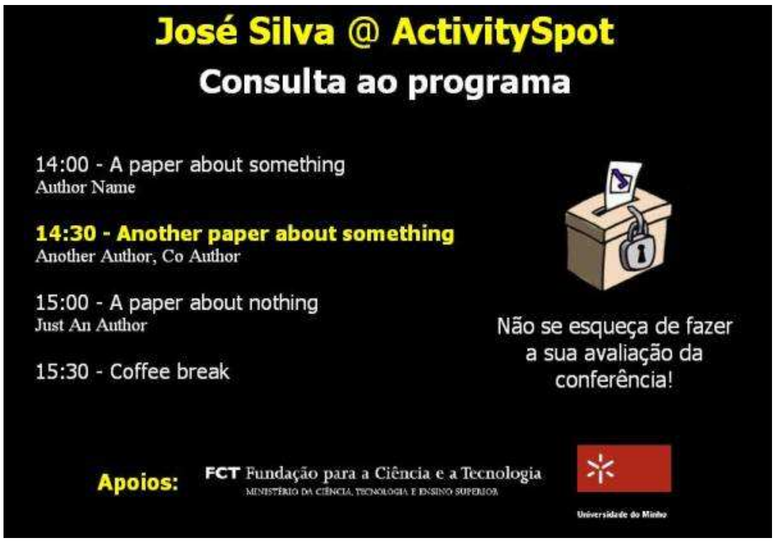
The interaction means available for this study comprised an SMS gateway and two interaction spots installed at the conference reception hall, where coffee breaks also took place. Each interaction spot was equipped with a computer running a public display (using a 17" LCD screen) and connected to an RFID reader and a Bluetooth sensor.

During the conference, participants were asked to enroll in the study, by registering at the ActivitySpot desk. This registration step lasted about a minute – just the time for asking the participant name, intended activity, mobile phone number, research interests, and delivering two RFID tags. If a participant was using a Bluetooth- and camera-enabled mobile phone, an additional step – obtaining automatically its Bluetooth MAC address – was required[2].

Proposed actions

The actions available for the supported activities were:

- viewing the conference program (for all activities) – at an interaction spot, participants swipe their RFID card over the reader and see the response in the LCD display. The response content includes the next three events in the conference program, with events matching personal research interests highlighted, as well as suggestions for executing other actions, based on the personal activity history (sample response in figure 5.2).



José Silva @ ActivitySpot
Consulta ao programa

14:00 - A paper about something
Author Name

14:30 - Another paper about something
Another Author, Co Author

15:00 - A paper about nothing
Just An Author

15:30 - Coffee break

Não se esqueça de fazer a sua avaliação da conferência!

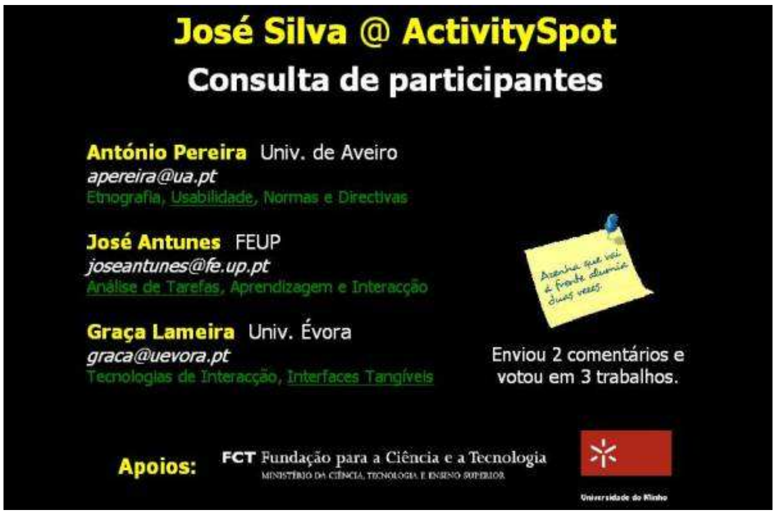
Apoios: FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
Universidade do Minho

Figure 5.2: A sample response for the conference program view action

[2]Unlike previous studies, participants sending photographs over Bluetooth could be identified by the system

- commenting a paper or poster (for all activities) – participants send an SMS with the syntax *comentar <paper or poster id> <comment>*. Papers and posters have a unique identifier. The response to this action is a confirmation message. The comment is immediately delivered to the paper or poster main author through SMS.

- viewing the participant list (for all activities) – at an interaction spot, participants swipe their RFID keyring over the reader and see the response in the LCD display. The response content includes a random selection of three participants (participants matching personal research interests have more chances to be presented and matching interests are underlined) and a summary of past executed actions (sample response in figure 5.3).



José Silva @ ActivitySpot
Consulta de participantes

António Pereira Univ. de Aveiro
apereira@ua.pt
Etnografia, Usabilidade, Normas e Directivas

José Antunes FEUP
joseantunes@fe.up.pt
Análise de Tarefas, Aprendizagem e Interacção

Graça Lameira Univ. Évora
graca@uevora.pt
Tecnologias de Interacção, Interfaces Tangíveis

Enviou 2 comentários e votou em 3 trabalhos.

Apoios: FCT Fundação para a Ciência e a Tecnologia MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Universidade do Minho

Figure 5.3: A sample response for the participant list view action

- rating a paper or poster (not available to conference organizers) – participants send an SMS with the syntax *votar <paper or poster id> <a value between 1 and 5>*. Each participant is allowed to rate each paper or poster only once. The response to this action is a confirmation message.

- sharing a photograph (for all activities) – participants owning a camera- and Bluetooth-enabled mobile phone send a photograph to the Bluetooth sensor located at an interaction spot. The response to this action – a confirmation message along with the shared photograph – is presented in the LCD display.

- rating a conference day (not available to conference organizers) – participants send an SMS with the syntax *avaliar <a value between 1 and 5>*. Each participant is allowed to rate each conference day only once. The rating is assigned to the conference day corresponding to the SMS reception date. The response to this action is a confirmation message.
- check paper or poster ratings (available only to authors) – participants send an SMS with the syntax *consultar*. The response to this action is a message with information about the rating average earned by the author's work. Authors could only check their work rating after having rated at least another work.
- check conference ratings (available only to conference organizers) – participants send an SMS with the syntax *consultar*. The response to this action is a message with information about the rating average earned by each conference day.
- broadcasting an advertisement (e.g., a change in the conference program) to conference participants (available only to conference organizers) – participants send an SMS with the syntax *avisar <advertisement text>*. The response to this action is a confirmation message. The advertisement is immediately delivered through SMS to all participants registered at ActivitySpot.
- undoing the last action (for all activities) – participants send an SMS with the syntax *anular*. The last undoable action (only photograph sharing and paper/-poster/conference ratings) is then rolled back. The response to this action is a confirmation message.

5.1.4 Results

In all the three studies, ActivitySpot evaluators used a ubiquitous computing system without previous training or even previous awareness of it. The first contact with ActivitySpot was generally made after reading advertisements spread throughout the physical space where the activities were available, mainly near the interaction devices.

These advertisements contained short instructions about registration and device usage. Due to this approach of not hiring people to use the system, few people volunteered for using ActivitySpot, when compared to the universe of visitors in each scenario.

All evaluators responded a survey, composed mainly of 4-point Likert scale answers (1 – totally disagree – to 4 – totally agree). The option for an even number of possible answers was made to reduce ambiguity and make participants definitely adopt a position instead of hiding themselves within an intermediary, uncommitted answer. In order to simplify the presentation and analysis of results, responses are aggregated into two categories: positive answers, i.e., meeting evaluation goals, and negative answers. I consider that a particular goal premise is met when the number of positive answers is above the third quartile. The statistical significance of the results is assessed by a Chi-square test attempting to reject, for each question, the null hypothesis that positive and negative answers had equal proportions, with at least a 95% confidence interval. I next describe the results for each premise of each evaluation goal (please, refer to the introduction to this section for the details of premises), mentioning the proportion of positive answers and respective Chi-square results[3], and conclude the sub-section with complementary remarks. The documentation on the online repository details how survey items relate to each evaluation goal.

Compatibility of the conceptual model

Table 5.1 summarizes the results for each premise composing this evaluation goal (the results for each study are numbered from 1 – PhD poster session scenario – to 3 – conference scenario):

In all the three studies, participants clearly understood the assistance that was being offered to their activity. This result was particularly expressive in the last two studies (96% and 100% respectively, ρ<0.005). It also appears evident to participants that all interactions were integrated into their activity (100%, ρ<0.005, in the cultural center study, and 100%, in the conference study). There was trouble in evaluating the compat-

[3]When no ρ value is provided, this means that the null hypothesis could not be rejected, though this could in most cases be achieved with a larger sample.

Table 5.1: Results for the compatibility of the conceptual model

Premise	**Pos. (1)**	ρ **(1)**	**Pos. (2)**	ρ **(2)**	**Pos. (3)**	ρ **(3)**
C1	86%	< 0,01	96%	< 0,005	100%	< 0,005
C2	N/A	N/A	N/A	N/A	82%	< 0,05
C3	N/A	N/A	100%	< 0,005	100%	–

ibility of the conceptual model of the provided assistance with the participants' mental model of the same activity, because the first two scenarios offered activity structures that visitors normally were not used to deal with. For example, when someone goes to the cultural center, he is not used to vote for a show, publish a comment, or share a photograph. In the PhD poster session and cultural center studies, an activity analysis prior to the system implementation would not be of much value, because these are very simple activities. Evaluating this type of ubiquitous computing systems in a real scenario that totally meets the evaluation requirements is very difficult. Visitors are offered actions that, though being interesting and useful, are not part of the everyday structure of the particular activity. It seems that work activities are more suitable to achieve conceptual compatibility, as is the case of the conference study, where proposed actions were more compatible with the conventional conference activity structure (82%, ρ<0.05).

User interaction

Table 5.2 summarizes the results for each premise composing the user interaction evaluation goal. Due to variations in the setting and evaluation strategy in the three studies, premise I1 had to be divided into several sub-premises partially evaluated in the different studies:

- I1.SMS – how easy was using SMS;
- I1.RFID – how easy was using RFID;
- I1.display – how easy was interacting with public displays;

- I1.camera – how easy was using the mobile phone camera for sending pictures over Bluetooth or infra-red;
- I1.instructions – whether the provided instructions were sufficient for an understanding of the interaction with ActivitySpot;
- I1.help – whether ActivitySpot would be easier to use if more help was available;
- I1.easiness – how easy was using ActivitySpot in general.

Table 5.2: Results for user interaction

Premise	**Pos. (1)**	ρ **(1)**	**Pos. (2)**	ρ **(2)**	**Pos. (3)**	ρ **(3)**
I1.SMS	93%	< 0,005	100%	< 0,005	N/A	N/A
I1.RFID	57%	–	95%	< 0,005	N/A	N/A
I1.display	64%	–	N/A	N/A	N/A	N/A
I1.camera	33%	–	70%	–	N/A	N/A
I1.instructions	N/A	N/A	79%	< 0,005	75%	–
I1.help	N/A	N/A	63%	–	100%	–
I1.easiness	N/A	N/A	N/A	N/A	88%	–
I2	N/A	N/A	96%	< 0,005	75%	–
I3	N/A	N/A	96%	< 0,005	88%	–
I4	80%	< 0,025	N/A	N/A	100%	< 0,005

The choice of grounding user interaction on basic, everyday interaction devices seems suitable to a walk-up-and-use ubiquitous computing system such as ActivitySpot. Given their previous experience in using some of the technology the studies were based on, participants had no trouble in interacting with ActivitySpot without previous training, particularly using SMS. Furthermore, participants generally were satisfied with the provided usage instructions, even if these were written very concisely, and did not find the initialization procedure (at the registration desk) cumbersome (80%, ρ<0.025 in the poster session study and 100%, ρ<0.005 in the conference one).

Regarding predictability and response time, participants of the last two studies were satisfied (respectively 96%, ρ<0.005 and 75%). In the poster session study, this issue could not be evaluated, due to technical problems.

Participants in general considered that using a system like ActivitySpot does not distract them from the activity they are carrying out (96%, ρ<0.005 in the cultural center study and 88% in the conference one).

Usefulness

Table 5.3 summarizes the results for each premise composing the usefulness evaluation goal. Premise U1 is decomposed in the following sub-premises:

- U1.useful – ActivitySpot helps visitors achieving the goals for their activity;
- U1.conventional – ActivitySpot helps achieving goals better than conventional assistance;
- U1.mobile – ActivitySpot helps achieving goals better than an application in a mobile phone;
- U1.kiosk – ActivitySpot helps achieving goals better than an interactive kiosk.

Premise U2 is sub-divided into U2.personalized (participants recognize that the system provides them with personalized information) and U2.adequate (participants consider that ActivitySpot provided them with an adequate level of personalization).

We adopted different evaluation strategies in each study for this evaluation goal. In the first two studies, we inquired participants for their general satisfaction regarding the system. Participants were generally satisfied with their experience (86%, ρ<0.01 for the poster session study, and 88%, ρ<0.005, for the cultural center one) and considered it more interesting than if it was carried out without system support (87%, ρ<0.005, for the poster session study, and 88%, ρ<0.005, for the cultural center one).

In the conference study, we introduced a control group, that was used to compare satisfaction regarding goal completion between system users and non-users. However, due to low participation at the conference and low response rate, we could not collect

Table 5.3: Results for usefulness

Premise	**Pos. (1)**	**ρ (1)**	**Pos. (2)**	**ρ (2)**	**Pos. (3)**	**ρ (3)**
U1.useful	<u>86%</u>	< 0,01	<u>88%</u>	< 0,005	<u>75%</u>	–
U1.conventional	<u>87%</u>	< 0,005	<u>88%</u>	< 0,005	13%	–
U1.mobile	N/A	N/A	N/A	N/A	<u>82%</u>	< 0,05
U1.kiosk	N/A	N/A	N/A	N/A	<u>91%</u>	< 0,01
U2.personalized	N/A	N/A	<u>88%</u>	< 0,005	<u>75%</u>	–
U2.adequate	40%	–	33%	–	38%	–

enough responses from the control group to obtain statistical significance. Therefore, we restricted usefulness evaluation in this study to the experimental group. Results showed that participants considered that the system support helped them in achieving the goals for their activity (75%), and that it was more effective than if it was provided over a single application on a mobile phone (82%, $\rho<0.05$) or an interactive kiosk (91%, $\rho<0.01$). However, participants interestingly stated that they could perfectly achieve their goals without ActivitySpot or any other computer system support (87%).

In the last two studies, all participants recognized that the system was providing them with personalized information (88%, $\rho<0.005$, for the cultural center study, and 75% for the conference one). However, the same participants considered that for personalization to be more useful, the system should have access to more personal data (about two thirds for all studies). This is an expected consequence of the current lack of solutions for the seamless integration between the local infrastructure and the personal domain.

Closing remarks

This series of user studies has allowed to demonstrate that visitors to public spaces can easily understand the type of activity-centered support provided by ActivitySpot and that they do not find obstacles in using the provided interaction means for carrying out their activity. Previous experience in using the basic interaction devices on which

ActivitySpot is grounded was fundamental for these results. However, it is not always possible to provide an activity model compatible with the visitors mental model, due to the nature of the activity itself, which, with the introduction of pervasive computing support, may become somehow artificial. This, along with the more or less compelling assistance that may be provided, which does not depend on the ActivitySpot infrastructure, may affect usefulness of the system. As noted by Edwards et al. [Edwards 03], infrastructures can only be evaluated in the context of use and thus must be evaluated indirectly through applications built on top of it, thus incurring in the risks of supporting unattractive applications or getting distracted by the demands of application development and to lose sight of the real purpose of the effort, which is purely to evaluate the infrastructure. Finally, usefulness is also influenced by the current lack of mechanisms for automatic integration between the local infrastructure and the visitors' domain, key to providing more effective personalization.

System usage log analysis and some observations provided some additional intriguing results:

- Participants tended to interact predominantly with the public displays, mainly with RFID tags (half of the interactions in the cultural center study and more than 75% in the conference study), probably due to ease of use and immediacy of response.

- Some participants complained about the cost of SMS usage, which ultimately resulted in a barrier to usage. This may be due to the lower SMS habits of our population sample (around the 30s).

- The importance of entertainment and engagement in this kind of system, reflected by the notorious pleasure that some participants demonstrated when sharing their own photographs with the system and watching them being displayed in the public screens to all other people.

The implementation of the various user studies has also been very useful in assessing the flexibility and ease-of-use of the ActivitySpot development framework. In particular, the simplicity of the stimuli-response paradigm, has proven to be very adaptive to multiple interaction artefacts and situations. All that developers need to do is to implement an action controller interface and write the logic for processing the stimuli of a given type and for producing a response. Both stimuli and responses are wrapped by a device-independent abstraction, which facilitates programmatic manipulation and speeds up learning and development time. All the code for the several actions that composed the activities in the various scenarios was written in a very short time. Each action required approximately one or two hours to be implemented. Furthermore, functional operations (e.g., specific database queries) were reused between different actions, also contributing to a shorter development time.

Action developers do not have to care about low-level details of interaction devices neither about activity-level concerns. They just focus on implementing the reaction to stimuli events. Furthermore, as demonstrated by the diversity of user studies, the ActivitySpot framework allowed for the development of heterogeneous actions, some of them supporting multiple interaction devices.

5.2 ActivitySpot environment management with the GUI authoring tool

The evaluation of the ActivitySpot authoring tool assessed whether: a) its interaction model is compatible with the user's mental model; b) it eases configuration tasks, not having to think of low level issues such as how input or output are captured or produced; c) it allows for rapid configuration of an ActivitySpot environment; d) it allows configuration of new interesting applications; and e) it helps anticipating end-user interaction problems. The following sub-sections detail the evaluation method and the results that were obtained.

Method

The typical user of the ActivitySpot authoring tool is a public space facilities manager, likely having basic end-user experience with computer tools. Therefore, the population of the user study had to be selected among a universe of people that usually perform such management tasks or, at least, have some sensibility for those tasks. The user sample was thus selected mainly among university staff matching that profile. The user study was set up with 13 volunteers (8 men and 5 women), aged between 25 and 46, and with varying computer science and physical space/facilities management expertise.

The user study was composed of individual evaluation sessions, being carried out along two weeks. An evaluation session had the following structure:

I. introduction – volunteers were given a short tutorial about the tool (main concepts, interface structure, and usage). Next, they were introduced to the scenario with which they worked: modelling activities taking place at the university campus. The scenario was simulated and therefore users did not deal with real interaction devices. For evaluation purposes, a simulated scenario was sufficient. The scenario included pre-configured interaction devices (e.g., gesture recognition, public displays, RFID readers, etc.) and action controllers common to different possible activities taking place at the university campus. Examples of such actions were "requesting directions to somebody's office", "viewing the university restaurant menu", "make a complaint", etc. A pre-sketched building plan was also loaded into the tool.

II. activity analysis – users were asked to imagine one activity (and respective actions) that would be interesting to support with the available interaction devices. Each user would later use this activity description in further evaluation steps.

III. training – users were asked to use the tool to accomplish several tasks with different complexity levels. Completion time and task success were measured for every task. Users were also videotaped and the environment specifications they produced were saved for further analysis. The required training tasks were as follows:

   1. positioning five different interaction devices in the physical space;
   2. specifying an action: name, action controller, and two stimulus-response pairs;
   3. specifying a second action: name, action controller, one stimulus-response pair, and a time-based execution condition;

4. specifying an activity: name, associated actions – the actions specified in the previous tasks –, an action-specific parameter, and a time-based activity execution condition;
5. specifying a third action: name, action controller, and a single stimulus-response pair;
6. specifying a second activity: name, associated actions – second and third actions –, an action-specific parameter, and a role-based, action-specific execution condition;
7. adding an action to an activity;
8. removing an action from an activity;

IV. execution – users were asked to use the tool for designing the activity specification they had described in the activity analysis step. Completion time was measured, users were again videotaped and the resulting environment specification saved. Furthermore, the think-aloud technique was used during this evaluation step.

V. interview – users were interviewed for feedback regarding the tool. The interview also included a visual analysis of the designed activity specifications, in order to evaluate if the tool was helping users in identifying possible end-user interaction problems in the scenario they designed.

Results

In the activity analysis step, all the volunteers comfortably used the concepts of activity and action for designing a new activity. Everybody thought of activities that usually take place at the university and employed mainly SMS stimuli and responses for supporting the actions composing their activity.

The training tasks were mainly targeted at measuring task completion time. Although the results did not have a normal distribution – standard deviations and coefficients of variation were high –, all the users performed their tasks in a reasonably short time. Action specification had a mean time of 80 seconds, with second and third action specification tasks with shorter times. Activity specification had a mean time of 150 seconds, with the second activity specification task with a shorter time. Men and computer science expert groups achieved shorter task completion times. Table 5.4 details the training tasks results.

Table 5.4: Completion time for each training task T (in seconds), for each subject (S), with mean, standard deviation, and coefficient of variation

S	**M/F**	**Expert**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
1	M	yes	105	106	108	170	80	165	85	15
2	M	yes	53	105	43	154	57	89	18	6
3	M	yes	27	51	63	93	28	68	22	12
4	M	no	101	89	62	134	73	146	18	16
5	M	yes	73	99	67	143	39	121	12	13
6	M	yes	62	113	72	114	50	102	9	10
7	M	no	123	75	64	139	33	141	15	10
8	F	no	198	163	130	322	81	177	10	11
9	F	no	105	136	65	256	32	130	11	8
10	M	yes	62	78	78	145	67	203	9	12
11	F	yes	58	45	46	152	35	89	6	8
12	F	no	113	245	74	182	60	147	10	14
13	F	no	62	226	73	246	77	182	137	18
		μ	88	118	73	173	55	135	28	12
		σ	43,6	61,2	23,3	64,3	19,8	40,6	38,7	3,4
		CV	49,6%	52%	32,1%	37,1%	36,2%	30%	139%	29,3%

In the execution step, results did not follow again a normal distribution (see table 5.5). The fact that volunteers specified heterogeneous activities, with variations in the number of devices, actions and stimulus-response pairs employed in the activity, affects even more the statistical relevance of these results. However, all the volunteers could completely configure the activity they specified in the introductory step with a mean time below 8 minutes. Configuring in 8 minutes an activity that employs an average of 8 devices, 3 actions, 6 stimulus-response pairs, and 2 execution conditions, is incontestably a short time.

Table 5.5: Completion time for the execution step (in seconds), for each subject (S), with mean, standard deviation, and coefficient of variation

S	**M/F**	**Expert**	**Task**
1	M	yes	738
2	M	yes	372
3	M	yes	434
4	M	no	366
5	M	yes	384
6	M	yes	535
7	M	no	567
8	F	no	390
9	F	no	462
10	M	yes	436
11	F	yes	253
12	F	no	609
13	F	no	358
		μ	454
		σ	128
		CV	28,3%

These short configuration times demonstrate how simple the tool concepts are and the productivity such a tool can provide to ubiquitous computing solutions administrators. However, videotape observations and interviews reported some user interface issues that hindered the performance of some users. The user interface problems were due to the lack of experience and care in developing a usable interface and were not related to the activity-based conceptual model proposed by the tool.

5.3 Summary

The validation of this thesis was composed of three evaluation stages, each contributing to achieve one or more validation goals. The first stage – implementation of the ActivitySpot run-time infrastructure – demonstrated that the proposed model of activity and user interaction is implementable in an effective ubiquitous computing infrastructure.

The second stage, in which three end-user studies were carried out based on real ubiquitous computing scenarios, demonstrated that visitors to public spaces can easily understand the type of activity-centered support provided by ActivitySpot and that they do not find obstacles in using the provided interaction means for carrying out their activity. Previous experience in using the basic interaction devices on which ActivitySpot is grounded was fundamental for these results. This stage also evaluated how easy and rapid it is to develop the support for new activities and actions. All the code for the several actions that composed the activities in the various scenarios was written in a very short time. Each action required approximately one or two hours to be implemented. Action developers do not have to care about low-level details of interaction devices neither about activity-level concerns. They just focus on implementing the reaction to stimuli events.

The final stage – a user study with the GUI authoring tool – demonstrated that the proposed model is fundamental in facilitating the task of public space administrators in deploying and managing a ubiquitous computing infrastructure and supported activities. All the subjects performed configuration tasks with ease and in a reasonably short time, despite some usability problems in the tool, that were due mainly to my lack of experience in developing GUI tools.

Chapter 6

Conclusions

A major challenge to ubiquitous computing system designers is the provision of walk-up-and-use solutions for supporting activities performed by occasional visitors to a particular place. When arriving for the first time to a particular place, occasional visitors have little or no idea about what the local environment is providing to support their activity. Furthermore, this support has to be self-explainable and quickly learnable, as occasional visitors are not prepared to interact with an unknown system and do not have time to spend understanding and learning how to use new tools.

Although ubiquitous computing has the potential for greatly enhancing the experience of occasional visitors to public places, there is still much to do to achieve the vision of a computing system that requires little or no attention at all, so that humans can use the computer unconsciously. Ideally, people should perform an activity requiring computing tools as they perform any other activity, by focusing on the activity itself, and using the computing tool as naturally as other tools.

This work follows an activity-centered approach to ubiquitous computing and has as main goal the development of ActivitySpot, an activity-centered conceptual and software framework targeted at supporting occasional visitors to public spaces. The conceptual framework, described in chapter 3, is intended to model human activity and user interaction with the ubiquitous computing system. Undertaking an activity-centered approach to ubiquitous computing system design requires an understanding of how humans think about and carry out their activities. Therefore, this research is grounded on previous work on human activity analysis, namely Activity Theory, a conceptual framework for analyzing human activity developed during the twentieth century. Activity Theory has matured along several decades and provides a set of simple and solid concepts, particularly those related to the structure of human activity (the levels of activity, actions, and operations), which form the basis of the activity model I propose.

For modelling user interaction, I assumed that interaction media are elementary ones, such as voice input/output, gesture recognition, RFID tag reading, SMS, public screen display, etc. Given that user interaction with a ubiquitous computing system is done through multiple, heterogeneous means and, in many cases, with little common characteristics, I reduced user interaction analysis to basic human-computer interaction concepts: stimulus and response. I assume that, for a given stimulus through a given interaction medium, a response is produced through the same medium or through other medium or set of media.

The software framework, described in chapter 4, includes a ubiquitous computing infrastructure for providing the actual support to occasional visitors, tools for deploying ubiquitous computing solutions by non-computer-expert public space administrators, and a software library for developing the support to new activities.

Chapter 5 details the method and the steps employed for validating my thesis. The ActivitySpot framework is validated from three perspectives: the end-user perspective (actual visitors interacting with the system in a public space); the manager perspective (authoring tool users); and the developer perspective (software library users). Three end-user studies with the ActivitySpot infrastructure deployed in real scenarios and a user study with the authoring tool provided the means for successfully completing the thesis validation.

6.1 Contributions

This work contributes to the area of ubiquitous computing by proposing a conceptual and software framework that facilitates the development and deployment of effective walk-up-and-use ubiquitous computing solutions targeted at supporting activities performed by occasional visitors.

6.1.1 Contributions of the conceptual framework

The conceptual framework is composed of an activity model and a user interaction model aiming at easing and enhancing the activity of occasional visitors to public spaces. The conceptual framework offers several theoretical contributions. It provides researchers with a concrete and practical application of Activity Theory concepts, enriching the corpus of experimentation with Activity Theory with an original approach. This approach, though centered on the aspect of activity structure, explores it deeply enough to provide practitioners with a relevant reference. The concepts of activity structure are explored both from software engineering and user interaction perspectives.

The activity-centered character of the proposed framework emphasizes the aspects that are closer to the common user perception of activity. Presenting visitors with a model of activity that is centered on the motive for their visit and on the actions they can execute to carry out that activity, rather than the usual model based on applications, provides significative advantages in terms of rapid perception of the support provided by the ubiquitous computing environment. This contribution is backed by the results of the end-user studies, which showed that the ActivitySpot framework is effective for walk-up-and-use systems, turning user interaction with a ubiquitous computing system almost as natural as interacting with other everyday tools. The majority of users clearly reported that ActivitySpot fostered learnability and usability. The choice of using elementary, everyday interaction means with a simple stimulus-response interaction model was also fundamental in the success with end-users.

From a software engineering perspective, the structural aspect of activity is a catalyzer for modularity and reuse, in line with fundamental software engineering practices and patterns. Actions are seen as loosely-coupled units of activity with self-contained behavior, implementing a simple contract based on reactions to stimuli and generation of responses, which enables action reuse among multiple activities. Furthermore, by decoupling activity logic or, more precisely, operational logic from the interaction devices used by visitors, i.e., separating input, logic, and output, the ActivitySpot framework facilitates the seamless substitution of interaction possibilities as well as the introduction of new, unanticipated devices, thus supporting evolution. The development of the support for the several scenarios where end-user studies took place demonstrated the impact in productivity and software manageability brought by action/operation reuse and separation of logic from interaction. The implementation of that support was achieved in a few days of work for each scenario, a critical factor in any software development initiative.

6.1.2 Contributions of the software framework

The software framework is composed of a run-time infrastructure implementing the conceptual model, an authoring tool for public space managers, and a software library for developers. All these components showed that the conceptual framework can be realized into a concrete, end-to-end implementation of a ubiquitous computing framework.

The run-time infrastructure successfully ran several end-user studies, with varying usage loads and diverse, heterogeneous interaction means. The infrastructure requires only a Java virtual machine environment and a Java-based tuple-space, along with gateways for each interaction device. It does not avoid the chores of ubiquitous computing device deployment, but strongly alleviates the challenge of user interaction coordination and activity management. Due to its lightweight deployment and management characteristics, the ActivitySpot infrastructure was used in several occasions outside the scope of this work, such as public events at the University of Minho or technology fairs.

The GUI authoring tool was evaluated by several users, demonstrating that it facilitates the configuration of the support to ActivitySpot-enabled environments. This tool is based on the proposed conceptual model, representing activities, actions, interaction devices, and physical space. Its users can assign actions to activities, devices to actions, and represent device distribution over the physical space. Further configuration is also possible, such as defining context-based execution conditions for activities and actions. With this tool, public space managers with common computer knowledge can easily configure ActivitySpot-enabled environments without having to care about the internals of the infrastructure or about the details of user interaction devices.

The software library was designed exclusively for supporting the development of new actions. It abstracts user interaction device details and does not tie developers to the activities the actions are going to be part of. Developers only have to care about writing the code for stimulus reaction behavior, using abstractions for representing stimuli and responses. With this library, any developer with minimum Java experience can develop the support for new actions. Moreover, its simplicity and focus on stimulus-response processing enables rapid development, being decisive in the short development timings achieved during the preparation of end-user studies and other scenarios.

All the ActivitySpot software components are available [Pinto 08] to researchers and ubiquitous computing practitioners willing to build upon it. The ubiquitous computing research area has achieved a stage in its evolution when building entire solutions from scratch hardly brings relevant contributions and is a waste of resources. By facilitating rapid development of ubiquitous computing solutions, the ActivitySpot framework enables researchers to focus on higher-level concerns, such as social or psychological aspects.

6.2 Limitations

There is no research work without limitations and my work is not the exception. Some limitations are a result of the need of scoping that is inherent to any research activity while others are a result of the lack of time to complete particular research tasks.

The major limitation of my work is the solidity and completeness of the validation process. This work originally intended to thoroughly validate each element of the ActivitySpot framework – from the conceptual model to the software components. The major effort investment was put on validating the conceptual model and the software infrastructure from the end-user viewpoint. Three user studies were successfully carried out in distinct and realistic scenarios, producing interesting and meaningful results, which allowed me to partially validate my thesis: the conceptual model is compatible with the end-user mental model and the approach to user interaction, based on the composition of elementary interaction means, was successfully apprehended by end-users. However, the method employed in the user studies had some variations, from study to study, which affected the solidity of the results. For example, for user interaction evaluation, only a single evaluation criterion was shared between the first and third studies – the remaining criteria for that evaluation goal were different between these two studies. These variations were introduced after the first and second studies, as I was learning with some previous errors. Moreover, in some studies – particularly the first and third ones –, the user sample was short, avoiding the obtention of statistical significance in the results. In summary, the end-user studies would benefit of a larger sample and a validation method applied consistently along the studies, which would lead to more solid and consistent results.

Regarding the authoring tool, evaluation has been negatively affected by the primitivity of the user interface and by the confusion provoked by the existence of two editors with a physical space representation – space editor and activities editor. Moreover, the user sample was short, as proven by the high standard deviations and coefficients of variation in the results. Otherwise, the evaluation process was consistent between all the user study sessions.

The authoring tool would increase even more the productivity if it had a deeper integration with the run-time infrastructure. After specifying an environment configuration, users had to manually copy the generated specification to a specific folder in the run-time infrastructure installation. This is not at all a user-friendly task. A one-click automated deployment functionality in the authoring tool would surely be a much better solution.

I did not invest in the evaluation of the software library for ubiquitous computing developers, because it had a secondary importance compared to other components of the framework. A rigorous evaluation process would try to assess how the software library augments productivity, whether it is learnable and easy to use, and whether it responds to a wide diversity of scenarios. The evaluation sample would have to include a reasonable number of more or less experienced developers. My own use of the library somehow demonstrated these points, though admittedly in a biased way.

Finally, although ActivitySpot was deployed and positively evaluated in real scenarios, it has not yet been employed in other real settings. Further publication of these results and collaboration with the industry may contribute to concretize this goal.

6.3 Future work

The ubiquitous computing scenarios that this work targets are very specific: occasional, simple activities executed by individuals in public places, using elementary ubiquitous computing means. Other related approaches may be the subject of relevant future work. For example, enlarging the scope of my approach to workplace scenarios, where activities are recurrently executed by the same people. Workplace scenarios would assess whether the ActivitySpot framework is effective in situations where the right support for repetitive and cumbersome tasks is fundamental. Moreover, it is easier to assess whether the ActivitySpot conceptual model fits people's model of a work activity, because people are more able to externalize their activity – thinking about what they do – than they are with activities they perform occasionally. Workplace scenarios would also be an opportunity for exploring how the ActivitySpot framework can deal with collaborative activities: dealing with multiple interaction means shared by multiple users working on the same activity brings many challenges not covered by my work.

The application of the fundamental principles of Activity Theory to my work was partial; ActivitySpot is centered on the structural aspect of Activity Theory, i.e., how an activity is decomposed into actions and operations. Other aspects, such as goal formation, externalization-internalization, tool mediation, or cultural-historical development (see section 3.1.1) are not explicitly represented by my conceptual model and can be subject for future work. Goal formation and cultural-historical development are subjects of particular relevance to ubiquitous computing.

The principle of goal formation states that the consciousness of the subject regarding motive and goals within an activity is dynamic and that different subjects may see an activity or action differently, depending on their motive and goals. For example, an action may become an activity by itself, when it acquires enough relevance to serve a motive. For example, procuring ingredients is an action for a domestic cook preparing the dinner for his family, but can become an activity in its own if we think of the procurement department in a restaurant chain. The inverse phenomenon may also happen when an activity loses the motive that inspired it, whereupon it is converted into an action (with its own goal) that implements a different activity. Therefore, when designing ubiquitous computing support for human activities, one has to think of the different motives and goals a particular functionality may serve, even for the same person. The biggest challenge here is how to introduce flexibility in a ubiquitous computing system so that a particular functionality is perceived and executed differently by different people, depending on their motive.

Cultural-historical development of activity is also a matter of particular relevance. Activities evolve over time, influenced by cultural and historical forces. People constantly look for improving the way a particular activity is executed and may (or not) share this knowledge with others. Although the act of sharing the praxis of an activity with other people does not seem a major challenge for ubiquitous computing, the capability of integrating feedback into the way an activity or action is executed is not trivial and deserves to be the subject of future work.

References

[Abowd 97] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper & M. Pinkerton. *Cyberguide: A mobile context-aware tour guide.* Wireless Networks, vol. 3, no. 5, pages 421–433, 1997.

[Abowd 00] Gregory D. Abowd & Elizabeth D. Mynatt. *Charting Past, Present, and Future Research in Ubiquitous Computing.* ACM Transactions on Computer-Human Interaction, vol. 7, no. 1, pages 29–58, March 2000.

[Alexander 77] C. Alexander, S. Ishikawa & M. Silverstein. A pattern language: Towns, buildings, and construction. Oxford University Press, New York, USA, 1977.

[Banavar 00] Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman & Deborra Zukowski. *Challenges: An Application Model for Pervasive Computing.* In Sixth ACM/IEEE International Conference on Mobile Networking and Computing, pages 266–274, Boston, USA, August 2000. ACM Press.

[Bannon 91] L. Bannon & S. Bødker. *Beyond the Interface: Encountering Artifacts in Use.* In J. Carroll, editeur, Designing Interaction: Psychology at the Human-Computer Interface, pages 227–253. Cambridge University Press, New York, USA, 1991.

[Bardram 97] J. Bardram. *Plans as Situated Action: An Activity Theory Approach to Workflow Systems.* In Fifth European Conference on Computer

Supported Cooperative Work (ECSCW '97), pages 17–32, Lancaster, United Kingdom, September 1997. Kluwer Academic Publishers.

[Beckmann 03] C. Beckmann & A. Dey. *SiteView: Tangibly Programming Active Environments with Predictive Visualization.* Technical Report IRB-TR-03-019, Intel Research, Berkeley, CA, USA, June 2003.

[Bellotti 02] V. Bellotti, M. Back, W. Edwards, R. Grinter, A. Henderson & C. Lopes. *Making Sense of Sensing Systems: Five Questions for Designers and Researchers.* In ACM Conference on Human Factors in Computing Systems (CHI 2002), pages 415–422, Minneapolis, MN, USA, April 2002.

[Bellotti 03] V. Bellotti, N. Ducheneaut, M. Howard & I. Smith. *Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool.* In ACM Conference on Human Factors in Computing Systems (CHI 2003), pages 345–352, Fort Lauderdale, USA, April 2003.

[Blackwell 01] A. Blackwell & R. Hague. *AutoHAN: An Architecture for Programming the Home.* In 2001 IEEE Symposia on Human-Centric Computing Languages and Environments (HCC '01), pages 150–157, Stresa, Italy, September 2001. IEEE Computer Society.

[Bødker 91a] S. Bødker. *Activity Theory as a Challenge to Systems Design.* In H.-E. Nissen, H. K. Klein & R. Hirscheim, editeurs, Information Systems Research: Contemporary Approaches & Emergent Traditions, pages 551–564. North Holland, Amsterdam, Netherlands, 1991.

[Bødker 91b] S. Bødker. Through the interface: A human activity approach to user interface design. Lawrence Erlbaum Associates, Hillsdale, USA, 1991.

[Card 83] S. Card, T. Moran & A. Newell. The psychology of human-computer interaction. Lawrence Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

[Cheverst 00] K. Cheverst, N. Davies, K. Mitchell & A. Friday. *Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project.* In 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000), pages 20–31, Boston, USA, August 2000. ACM Press.

[Christensen 02] H.B. Christensen & J. Bardram. *Supporting Human Activities – Exploring Activity-Centered Computing.* In Fourth International Conference on Ubiquitous Computing (UbiComp 2002), volume 2498, pages 107–116, Göteborg, Sweden, September 2002. Springer-Verlag.

[Compeau 95] D. Compeau & C. Higgins. *Computer Self-Efficacy: Development of a Measure and Initial Test.* MIS Quarterly, vol. 19, no. 2, pages 189–211, 1995.

[Constantine 99] L. Constantine & L. Lockwood. Software for use: A practical guide to the essential models and methods of usage-centered design. Addison-Wesley, Reading, MA, USA, 1999.

[Constantine 06] L. Constantine. *Activity Modeling: Toward a Pagmatic Integration of Activity Theory with Usage-Centered Design.* Technical paper, Laboratory for Usage-centered Software Engineering, Constantine & Lockwood. Ltd., November 2006.

[Cwalina 05] K. Cwalina & B. Abrams. Framework design guidelines: Conventions, idioms, and patterns for reusable .NET libraries. Addison Wesley, Reading, MA, USA, 2005.

[de Ipiña 02] D. Lpez de Ipiña, P. Mendonça & A. Hopper. *TRIP: a Low-Cost Vision-Based Location System for Ubiquitous Computing.* Personal and Ubiquitous Computing, vol. 6, no. 3, pages 206–219, May 2002.

[Dey 99] A. Dey, D. Salber, G. Abowd & M. Futakawa. *The Conference Assistant: Combining Context-Awareness with Wearable Computing.* In 3rd

International Symposium on Wearable Computers (ISWC '99), pages 21–28, San Francisco, USA, October 1999.

[Dey 04] A. Dey, R. Hamid, C. Beckmann, I. Li & D. Hsu. *a CAPpella: Programming by Demonstration of Context-Aware Applications.* In ACM Conference on Human Factors in Computing Systems (CHI 2004), pages 33–40, Vienna, Austria, April 2004.

[Dey 06] A. Dey, T. Sohn, S. Streng & J. Kodama. *iCAP: Interactive Prototyping of Context-Aware Applications.* In Fourth International Conference on Pervasive Computing (Pervasive 2006), pages 254–271, Dublin, Ireland, May 2006.

[Diaper 03] D. Diaper & N. Stanton. The handbook of task analysis for human-computer interaction. CRC, Boca Raton, FL, USA, 2003.

[Driver 04] C. Driver & S. Clarke. *Hermes: A Software Framework for Mobile, Context-Aware Trails Applications.* In J. Bardram, H. Christensen, D. Garlan & J. Sousa, editeurs, First International Workshop on Computer Support for Human Tasks and Activities, Vienna, Austria, April 2004.

[Duignan 06] M. Duignan, J. Noble & R. Biddle. *Activity Theory for Design: From Checklist to Interview.* In T. Clemmensen, P. Campos, R. Orngreen, A. Pejtersen & W. Wong, editeurs, Human Work Interaction Design: Designing for Human Work – The First IFIP TC 13.6 WG Conference (HWID '06), pages 9–32, Madeira, Portugal, February 2006. Springer-Verlag.

[Edwards 03] W.K. Edwards, V. Bellotti, A. Dey & M. Newman. *Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Middleware.* In 2003 Conference on Human Factors in Computing Systems (CHI 2003), Fort Lauderdale, USA, April 2003.

[Engeström 87] Y. Engeström. Learning by expanding: An activity-theoretical approach to developmental research. Orienta-Konsultit, Helsinki, Finland, 1987.

[Engeström 99a] Y. Engeström. *Activity theory and individual and social transformation.* In Y. Engeström, R. Miettinen & R.-L. Punamäki, editeurs, Perspectives on Activity Theory, pages 19–38. Cambridge University Press, Cambridge, UK, 1999.

[Engeström 99b] Y. Engeström, R. Miettinen & R.-L. Punamäki. Perspectives on activity theory. Cambridge University Press, Cambridge, UK, 1999.

[Eustice 99] K. Eustice, T. Lehman, A. Morales, M. Munson, S. Edlund & M. Guillen. *A Universal Information Appliance.* IBM Systems Journal, vol. 38, no. 4, pages 575–601, October 1999.

[Feiner 97] S. Feiner, B. MacIntyre, T. Höllerer & A. Webster. *A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment.* In First International Symposium on Wearable Computers (ISWC 97), pages 74–81, Cambridge, MA, USA, October 1997.

[Fitzmaurice 95] G. Fitzmaurice, H. Ishii & W. Buxton. *Bricks: Laying the Foundations for Graspable User Interfaces.* In ACM Conference on Human Factors in Computing Systems (CHI '95), pages 442–449, Denver, CO, USA, May 1995.

[Fleck 02] M. Fleck, M. Frid, T. Kindberg, E. O'Brien-Strain, R. Rajani & M. Spasojevic. *From Informing to Remembering: Ubiquitous Systems in Interactive Museums.* IEEE Pervasive Computing, vol. 1, no. 2, pages 13–21, April-June 2002.

[Gajos 02] K. Gajos, H. Fox & H. Shrobe. *End User Empowerment in Human Centered Pervasive Computing.* In Friedemann Mattern & Mahmoud Naghshineh, editeurs, First International Conference on Pervasive Computing (Pervasive 2002), pages 134–140, Zürich, Switzerland, August 2002. Springer-Verlag.

[Georgakopoulos 95] D. Georgakopoulos, M. Hornick & A. Sheth. *An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure.* Journal of Distributed and Parallel Databases, vol. 3, no. 2, pages 119–152, April 1995.

[Greenhalgh 02] C. Greenhalgh. *EQUIP: a Software Platform for Distributed Interactive Systems.* Technical Report Equator-02-002, Department of Computer Science, University of Nottingham, 2002.

[Greenhalgh 04] C. Greenhalgh, S. Izadi, J. Mathrick, J. Humble & I. Taylor. *ECT: A Toolkit to Support Rapid Construction of Ubicomp Environments.* In System Support for Ubiquitous Computing Workshop at the Sixth International Conference on Ubiquitous Computing (UbiComp 2004), Nottingham, England, September 2004.

[Hall 66] E. Hall. The hidden dimension. Doubleday, New York, USA, 1966.

[Harrison 96] S. Harrison & P. Dourish. *Re-place-ing space: the roles of place and space in collaborative systems.* In 1996 ACM Conference on Computer Supported Cooperative Work (CSCW 1996), pages 67–76, Boston, MA, USA, November 1996. ACM Press.

[Hartson 92] R. Hartson & P. Gray. *Temporal Aspects of Tasks in the User Action Notation.* Human Computer Interaction, vol. 7, no. 1, pages 1–45, 1992.

[Henderson 86] D. A. Henderson & S. Card. *Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface.* ACM Transactions on Graphics, vol. 5, no. 3, pages 211–243, July 1986.

[Hess 02] Christopher K. Hess, Manuel Román & Roy H. Campbell. *Building Applications for Ubiquitous Computing Environments.* In Friedemann Mattern & Mahmoud Naghshineh, editeurs, First International Conference on Pervasive Computing (Pervasive 2002), pages 16–29, Zürich, Switzerland, August 2002. Springer-Verlag.

[Hodes 98] T. D. Hodes & H. Katz Randy. *Enabling "smart spaces": entity description and user interface generation for a heterogeneous component-based distributed system.* Technical Report CSD-98-1008, University of California at Berkeley, July 1998.

[Hong 00] J. Hong & J. Landay. *SATIN: A Toolkit for Informal Ink-based Applications.* In 13th Annual ACM Symposium on User Interface Software and Technology, pages 63–72, San Diego, CA, USA, November 2000.

[Humble 03] J. Humble, A. Crabtree, T. Hemmings, K. Åkesson, B. Koleva, T. Rodden & P. Hansson. *"Playing with the Bits": User-configuration of Ubiquitous Domestic Environments.* In Fifth International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, USA, October 2003.

[Jacob 86] R. Jacob. *A Specification Language for Direct Manipulation Interfaces.* ACM Transactions on Graphics, vol. 5, no. 4, pages 283–317, October 1986.

[Johanson 02] B. Johanson, A. Fox & T. Winograd. *The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms.* IEEE Pervasive Computing, vol. 1, no. 2, pages 67–74, April-June 2002.

[José 03] R. José, Helder Pinto, F. Meneses, N. Vilas Boas, H. Rodrigues & A. Moreira. *System support for Integrated Ubiquitous Computing Environments.* In System Support for Ubiquitous Computing Workshop at the Fifth International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, USA, 12-15 October 2003.

[José 04] R. José & P. Coutinho. *Situated web portal for local awareness and transient interaction.* In 2nd International Workshop on Ubiquitous Systems for Supporting Social Interaction and Face-to-Face Communication in Public Spaces at the Sixth International Conference on Ubiquitous Computing (UbiComp 2004), Nottingham, United Kingdom, September 2004.

[Kaptelinin 99] V. Kaptelinin, B. Nardi & C. Macaulay. *The Activity Checklist: A Tool for Representing the "Space" of Context.* Interactions, vol. 6, no. 4, pages 27–39, July-August 1999.

[Kaptelinin 03] V. Kaptelinin. *UMEA: Translating Interaction Histories into Project Contexts.* In G. Cockton & P. Korhonen, editeurs, ACM Conference on Human Factors in Computing Systems (CHI 2003), pages 353–360, Fort Lauderdale, USA, April 2003.

[Kindberg 00] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Ilja Bedner, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, Celine Pering, John Schettino & Bill Serra. *People, Places, Things: Web Presence for the Real World.* In 3rd Annual Conference on Wireless and Mobile Computer Systems and Applications (WMCSA 2000), Monterey, USA, December 2000.

[Ko 04] A. Ko, B. Myers & H. Aung. *Six Learning Barriers in End-User Programming Systems.* In IEEE Symposium on Visual Languages and Human-Centric Computing, pages 199–206, Rome, Italy, September 2004.

[Koile 03] K. Koile, K. Tollmar, D. Demirdjian, H. Shrobe & T. Darrell. *Activity Zones for Context-Aware Computing.* In Fifth International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, USA, October 2003.

[Krasner 88] G. E. Krasner & S. T. Pope. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80.* Journal of Object-Oriented Programming, vol. 1, no. 3, pages 26–49, August-September 1988.

[Kray 04] C. Kray, R. Wasinger & G. Kortuem. *Concepts and issues in interfaces for multiple users and multiple devices.* In Workshop on Multi-User and Ubiquitous User Interfaces (MU3I) at the 2004 International Conference on Intelligent User Interfaces (IUI 2004), pages 7–11, Funchal, Portugal, January 2004.

[Kules 03] B. Kules, H. Kang, C. Plaisant, A. Rose & B. Shneiderman. *Immediate Usability: Kiosk design principles from the CHI 2001 Photo Library.* Technical Report HCIL-2003-22, University of Maryland, January 2003.

[Kuutti 91a] K. Kuutti. *Activity Theory and its applications to information systems research and development.* In H.-E. Nissen, H. Klein & R. Hirschheim, editeurs, Information Systems Research: Contemporary Approaches & Emergent Traditions, pages 529–549. North-Holland, Amsterdam, Netherlands, 1991.

[Kuutti 91b] K. Kuutti. *The Concept of Activity as a Basic Unit for CSCW Research.* In L. Bannon, M. Robinson & K. Schmidt, editeurs, 2nd European Conference on Computer-Support Collaborative Work, pages 249–264, Amsterdam, Netherlands, 1991. Kluwer.

[Law 99] J. Law & J. Hassard. Actor network theory and after. Blackwell Publishers, Oxford, UK, 1999.

[Leontiev 78] A. Leontiev. Activity, consciousness, and personality. Prentice-Hall, Englewood Cliffs, NJ, USA, 1978.

[Leontiev 81] A. Leontiev. *The Problem of Activity in Psychology.* In J. Wertsch, editeur, The Concept of Activity in Soviet Psychology, pages 37–71. Sharpe, Armonk, USA, 1981.

[Look 03] Gary Look, Stephen Peters & Howard Shrobe. *Plan-Driven Ubiquitous Computing.* In Artificial Intelligence in Mobile System Workshop at the Fifth International Conference on Ubiquitous Computing (UbiComp 2003), Seattle, USA, 12-15 October 2003.

[Luria 76] A. Luria. Cognitive development: Its cultural and social foundations. Harvard University Press, Cambridge, MA, USA, 1976.

[MacIntyre 01] B. MacIntyre, E. Mynatt, S. Voida, K. Hansen, J. Tullio & G. Corso. *Support for multitasking and background awareness using interactive peripheral displays.* In ACM Symposium on User Interface Software and Technology (UIST '01), pages 41–50, Orlando, USA, November 2001.

[Medina-Mora 92] R. Medina-Mora, T. Winograd, R. Flores & F. Flores. *The Action Workflow Approach to Workflow Management Technology.* In ACM Conference on Computer-Supported Cooperative Work (CSCW 1992), pages 281–288, Toronto, Canada, November 1992.

[Muller 04] M. Muller, W. Geyer, B. Brownholtz, E. Wilcox & D. Millen. *One-Hundred Days in an Activity-Centric Collaboration Environment based on Shared objects.* In ACM Conference on Human Factors in Computing Systems (CHI 2004), pages 375–382, Vienna, Austria, April 2004.

[Myers 00] B. Myers, S. Hudson & R. Pausch. *Past, Present and Future of User Interface Software Tools.* ACM Transactions on Computer-Human Interaction, vol. 7, no. 1, pages 3–28, March 2000.

[Myers 02] B. Myers, R. Malkin, M. Bett, A. Waibel, B. Bostwick, R. Miller, J. Yang, M. Denecke, E. Seemann, J. Zhu, C. Peck, D. Kong,

J. Nichols & B. Scherlis. *Flexi-modal and multi-machine user interfaces.* In Fourth IEEE International Conference on Multimodal Interfaces (ICMI 2002), pages 343–348, Pittsburgh, PA, USA, October 2002.

[Nardi 96] B. A. Nardi. Context and consciousness: Activity theory and human-computer interaction. MIT Press, Cambridge, USA, 1996.

[Newman 02] M. Newman, J. Sedivy, C. Neuwirth, W. Edwards, J. Hong, S. Izadi, K. Marcelo & T. Smith. *Designing for Serendipity: Supporting End-User Configuration of Ubiquitous Computing Environments.* In Symposium on Designing Interactive Systems (DIS 2002), pages 147–156, London, UK, June 2002.

[Norman 88] D. Norman. The psychology of everyday things. Basic Books, New York, USA, 1988.

[Norman 98] D.A. Norman. The invisible computer. MIT Press, Cambridge, USA, 1998.

[Norman 05] D. Norman. *Human-Centered Design Considered Harmful.* Interactions, vol. 12, no. 4, pages 14–19, July-August 2005.

[OASIS 03] OASIS. *Web Services for Remote Portlets (WSRP)*, 2003. http://oasis-open.org/committees/wsrp/.

[Paolucci 99] M. Paolucci, O. Shehory, K. Sycara, D. Kalp & A. Pannu. *A Planning Component for RETSINA Agents.* In M. Wooldridge & Y. Lesperance, editeurs, Lecture Notes in Artificial Intelligence, Intelligent Agents VI. MIT Press, Cambridge, USA, 1999.

[Paternò 99] F. Paternò. Model-based design and evaluation of interactive applications. Springer Verlag, London, UK, 1999.

[Paternò 01] F. Paternò. *Task Models in Interactive Software Systems.* In S. Chang, editeur, Handbook of Software Engineering & Knowledge Engineering, pages 817–836. World Scientific Publishing Co., Singapore, 2001.

[Pinto 05] H. Pinto & R. José. *Pervasive Location-based Systems: The Fundamental Challenges between Vision and Reality.* International Journal of Pervasive Computing and Communications, vol. 1, no. 1, pages 7–12, March 2005.

[Pinto 08] H. Pinto. *Activity-oriented Support to Localized Activities Performed by Occasional Visitors*, 2008. http://ubicomp.algoritmi.uminho.pt/activityspot/.

[Richardson 94] T. Richardson, F. Bennett, G. Mapp & A. Hopper. *Teleporting in an X Window System Environment.* IEEE Personal Communications, vol. 1, no. 3, pages 6–12, September 1994.

[Robertson 00] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel & V. Gorokhovsky. *The Task Gallery: A 3D Window Manager.* In ACM Conference on Human Factors in Computing Systems (CHI 2000), pages 494–501, Amsterdam, The Nederlands, April 2000.

[Rodden 03] T. Rodden & S. Benford. *The evolution of buildings and implications for the design of ubiquitous domestic environments.* In ACM Conference on Human Factors in Computing Systems (CHI 2003), pages 9–16, Fort Lauderdale, FL, USA, April 2003. ACM Press.

[Román 02a] M. Román & R. Campbell. *A User-Centric, Resource-Aware, Context-Sensitive, Multi-Device Application Framework for Ubiquitous Computing Environments.* Technical Report UIUCDCS-R-2002-2284 UILU-ENG-2002-1728, University of Illinois at Urbana-Champaign, July 2002.

[Román 02b] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell & K. Nahrstedt. *A Middleware Infrastructure for Active Spaces.* IEEE Pervasive Computing, vol. 1, no. 4, pages 74–83, October-December 2002.

[Satyanarayanan 01] M. Satyanarayanan. *Pervasive Computing: Vision and Challenges.* IEEE Personal Communications, vol. 8, no. 4, pages 10–17, August 2001.

[Schäl 96] T. Schäl. Workflow management systems for process organisations. Springer, Berlin, Germany, 1996.

[Scholtz 04] J. Scholtz & S. Consolvo. *Toward a Framework for Evaluating Ubiquitous Computing Applications.* IEEE Pervasive Computing, vol. 3, no. 2, pages 82–88, April-June 2004.

[Sierhuis 97] M. Sierhuis & W. Clancey. *Knowledge, Practice, Activities and People.* In AAAI Spring Symposium on Artificial Intelligence in Knowledge Management, pages 142–148, Stanford, USA, March 1997.

[Simmons 98] R. Simmons & D. Apfelbaum. *A Task Description Language for Robot Control.* In IEEE/RSJ International Conference on Intelligent Robotics and Systems, Vancouver, Canada, October 1998.

[SMSLib 08] SMSLib. *SMSLib*, 2008. http://smslib.org/.

[Sousa 02] J. P. Sousa & D. Garlan. *Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments.* In 3rd Working IEEE/IFIP Conference on Software Architecture, pages 29–43, Montreal, Canada, August 2002. Kluwer Academic Publishers.

[Sousa 05] J. P. Sousa. *Scaling Task Management in Space and Time: Reducing User Overhead in Ubiquitous-Computing Environments.* Phd thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, United States of America, May 2005.

[Stahl 05] C. Stahl, J. Baus, B. Brandherm, M. Schmitz & T. Schwartz. *Navigational and Shopping Assistance on the Basis of User Interactions in Intelligent Environments.* In The IEE International Workshop on Intelligent Environments (IE 2005), pages 182–191, Colchester, United Kingdom, June 2005.

[Suchman 87] L. Suchman. Plans and situated actions: the problem of human-machine communication. Cambridge University Press, New York, USA, 1987.

[Sun 03] Sun. *Enterprise JavaBeans Technology*, 2003. http://java.sun.com/products/ejb/.

[Tandler 01] Peter Tandler. *Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices.* In Third International Conference on Ubiquitous Computing (Ubicomp 2001), pages 96–115, Atlanta, Georgia, September 2001.

[Tolmie 02] P. Tolmie, J. Pycock, T. Diggins, A. MacLean & A. Karsenty. *Unremarkable computing.* In ACM Conference on Human Factors in Computing Systems (CHI 2002), Minneapolis, MN, USA, April 2002. ACM Press.

[Toye 05] E. Toye, R. Sharp, A. Madhavapeddy & D. Scott. *Using Smart Phones to Access Site-Specific Services.* IEEE Pervasive Computing, vol. 4, no. 2, pages 60–66, April-June 2005.

[Truong 04] K. Truong, E. Huang & G. Abowd. *CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home.* In Sixth International Conference on Ubiquitous Computing (UbiComp 2004), volume 3205, pages 143–160, Nottingham, England, September 2004. Springer-Verlag.

[Venkatesh 03] V. Venkatesh, M. Morris, G. Davis & F. Davis. *User Acceptance of Information Technology: Toward a Unified View.* MIS Quarterly, vol. 27, no. 3, pages 425–478, September 2003.

[Villar 05] N. Villar, G. Kortuem, K. Van Laerhoven & A. Schmidt. *The Pendle: A Personal Mediator for Mixed Initiative Environments.* In IEE Workshop Conference on Intelligent Environments (IE 05), Colchester, UK, June 2005.

[Vygotsky 78] L. Vygotsky. Mind in society: The development of higher psychological processes. Harvard University Press, Cambridge, MA, USA, 1978.

[W3C 06] W3C. *Extensible Markup Language (XML) 1.1 (Second Edition)*, 2006. http://www.w3.org/TR/2006/REC-xml11-20060816/.

[Want 02] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar & J. Light. *The Personal Server – Changing the Way We Think about Ubiquitous Computing.* In Fourth International Conference on Ubiquitous Computing (UbiComp 2002), pages 194–209, Göteborg, Sweden, October 2002. Springer-Verlag.

[Weiser 91] Mark Weiser. *The Computer for the 21st Century.* Scientific American, vol. 265, no. 3, pages 94–104, September 1991.

[Wertsch 81] J. Wertsch. *The Concept of Activity in Soviet Psychology: An Introduction.* In J. Wertsch, editeur, The Concept of Activity in Soviet Psychology, pages 3–36. Sharpe, Armonk, USA, 1981.

[Wise 00] A. Wise, A. Cass, B. Lerner, E. McCall, L. Osterweil & S. Sutton Jr. *Using Little-JIL to Coordinate Agents in Software Engineering.* In Fifteenth IEEE International Conference on Automated Software Engineering (ASE 2000), pages 155–163, Grenoble, France, September 2000.

Appendix A

Environment specification examples

This appendix complements chapter 4 with examples for each document defining an environment specification.

A.1 Devices specification

Listing A.1: An XML example of devices specification

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<devices>
 <device>
  <name>rfid</name>
  <type>input</type>
  <eventParameters>
   <parameter>
    <name>device-id</name>
    <type>string</type>
   </parameter>
   <parameter>
    <name>tag-id</name>
    <type>string</type>
   </parameter>
  </eventParameters>
  <controller>
   <binary>pt.uminho.activityspot.io.RFIDController</binary>
  </controller>
 </device>
 <device>
  <name>sms-out</name>
  <type>output</type>
  <eventParameters>
   <parameter>
    <name>destination</name>
    <type>string</type>
   </parameter>
   <parameter>
    <name>message</name>
    <type>string</type>
   </parameter>
  </eventParameters>
  <controller>
   <binary>pt.uminho.activityspot.io.SMSController</binary>
  </controller>
 </device>
</devices>
```

A.2 Context dimensions specification

Listing A.2: An XML example of context dimensions specification

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<context>
 <controller>
  <name>capability</name>
  <binary>
    pt.uminho.activityspot.context.CapabilityController
  </binary>
 </controller>
 <controller>
  <name>date-time</name>
  <binary>
    pt.uminho.activityspot.context.DateTimeController
  </binary>
 </controller>
</context>
```

A.3 Activities and actions specification

Listing A.3: An XML example of activities and actions specification

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<specification>
 <actions>
  <action id="c_1">
   <name>Get ticket</name>
   <description>
    Getting an electronic ticket
   </description>
   <controller>
    <binary>pt.uminho.activityspot.actions.GetTicket</binary>
   </controller>
   <stimuli>
    <stimulus>
     <type>rfid</type>
     <dataspace>event</dataspace>
     <response>
```

```
      <type multiple="false">sms-out</type>
     </response>
    </stimulus>
   </stimuli>
  </action>
  <action id="c_2">
   <name>Get friends list</name>
   <description>
    Obtaining a list of friend who have already bought ticket
   </description>
   <controller>
    <binary>pt.uminho.activityspot.actions.GetFriends</binary>
   </controller>
   <stimuli>
    <stimulus>
     <type>rfid</type>
     <dataspace>event</dataspace>
     <response>
      <type multiple="false">sms-out</type>
     </response>
    </stimulus>
   </stimuli>
  </action>
 </actions>
 <activities>
  <activity id="a_1">
   <name>Going to the cinema</name>
   <conditions>
    <condition type="date-time" value="19:00-24:00">
   </conditions>
   <actions>
     <action ref="c_1"/>
     <action ref="c_2"/>
   </actions>
  </activity>
 </activities>
 <devices>
  <device id="d_1">
   <type>sms-out</type>
   <location>l_1</location>
  </device>
  <device id="d_2">
   <type>rfid</type>
   <location>l_1</location>
  </device>
 </devices>
 <locations>
  <location id="l_1">
   <position>
    <x>711</x>
    <y>274</y>
   </position>
  </location>
 </locations>
</specification>
```

Appendix B

Evaluation materials

This appendix complements chapter 5 with the surveys submitted to the participants of each end-user evaluation scenario.

B.1 Phd poster session survey

Q1. Indique-nos qual foi a actividade que levou a cabo no sistema ActivitySpot.

◯ Visitar posters

◯ Apresentar posters

Q2. Antes de dar início à utilização do sistema ActivitySpot, tinha entendido o objectivo de cada uma das duas actividades propostas (ver questão anterior).

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q3. Os cartazes junto da recepção e os desdobráveis forneceram instruções claras para (marque uma cruz na casa indicada para cada linha):

	Concordo totalmente	Concordo parcialmente	Nem concordo, nem discordo	Discordo parcialmente	Discordo totalmente
Registo no sistema					
Reconhecer meios de interacção disponíveis					
Utilizar funcionalidades disponíveis					

Q4. O registo no sistema, antes de obter o cartão e o porta-chaves na recepção, foi realizado facilmente.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q5. Os meios de interacção disponibilizados pelo sistema foram de fácil utilização [Nota: caso não tenha utilizado algum dos meios abaixo listados, deixe a respectiva casa em branco]:

	Concordo totalmente	Concordo parcialmente	Nem concordo, nem discordo	Discordo parcialmente	Discordo totalmente
SMS					
Porta-chaves ou cartão					
Ecrãs públicos					
Página "Internet"					
Bluetooth ou infra-vermelhos					

Q6. A utilização da câmara fotográfica e a partilha de fotografias com o sistema tornou a experiência mais interessante [Nota: caso não tenha utilizado a câmara fotográfica para a partilha de fotografias com o sistema, não responda a esta questão]

○ Concordo totalmente
○ Concordo parcialmente
○ Nem concordo, nem discordo
○ Discordo parcialmente
○ Discordo totalmente

Q7. A sua experiência de utilização do ActivitySpot na "Semana da Engenharia" foi satisfatória.

○ Concordo totalmente
○ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q8. O suporte tecnológico à actividade que levou a cabo na "Semana da Engenharia" tornou-a mais interessante e envolvente do que se ela tivesse sido realizada através dos meios tradicionais.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q9. Acha que os seus anteriores conhecimentos de utilização de tecnologia contribuiram para o sucesso da experiência?

◯ Sim, sem dúvida

◯ Sim, provavelmente

◯ Talvez

◯ Provavelmente não

◯ Definitivamente não

Q10. Alguma vez durante a experiência executou alguma funcionalidade motivado pela curiosidade ou preocupou-se unicamente em levar a cabo a actividade?

◯ Sim, executei funcionalidades motivado(a) pela curiosidade

◯ Não me lembro

◯ Não, estive unicamente concentrado em conseguir executar apenas o que foi necessário.

Q11. O conhecimento que o sistema tinha do seu número de telemóvel e dos locais onde utilizou o cartão/porta-chaves não o preocupou.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q12. A experiência poderia ter sido mais interessante se o sistema tivesse acesso a mais informação pessoal (e.g., interesses de investigação).

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q13. A possível utilização futura para outros fins do conhecimento sobre o seu número de telemóvel e dos locais onde utilizou o cartão/porta-chaves não o preocupa.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Nem concordo, nem discordo

◯ Discordo parcialmente

◯ Discordo totalmente

Q14. Houve alguma interacção que não tenha dado origem a uma resposta do sistema?

◯ Sim, houve interacções sem resposta por parte do sistema

◯ Não me recordo

◯ Não, o sistema sempre respondeu às minhas interacções

Q15. Houve alguma interacção cuja resposta dada pelo sistema fosse desadequada ou desenquadrada do contexto?

◯ Sim, o sistema gerou respostas desadequadas/desenquadradas do contexto

◯ Não me recordo

◯ Não, o sistema sempre respondeu adequadamente às minhas interacções

Q16. O tempo decorrido entre as interacções e as respectivas respostas

geradas pelo sistema foi:

◯ Muito curto

◯ Curto

◯ Nem curto, nem longo

◯ Longo

◯ Muito longo

Q17. Caso tenha algum comentário ou sugestão adicionais em relação ao sistema ActivitySpot, por favor, use o espaço seguinte para o fazer:

____

____

____

____

Q18. Com que frequência utiliza o telemóvel para enviar SMS?

◯ Todos os dias

◯ Algumas vezes por semana

◯ Uma vez por semana

◯ Uma vez por mês

◯ Nunca

Q19. Com que frequência utiliza o computador?

◯ Todos os dias

◯ Algumas vezes por semana

◯ Uma vez por semana

◯ Uma vez por mês

◯ Nunca

Q20. Com que frequência utiliza o computador para navegar na "Internet"?

◯ Todos os dias

◯ Algumas vezes por semana

◯ Uma vez por semana

◯ Uma vez por mês

◯ Nunca

Q21. Com que frequência utiliza as funcionalidades Bluetooth ou infra-vermelhos do seu telemóvel? [Caso o seu telemóvel não possua, ou não sabe se possui, estas funcionalidades, não responda a esta questão]

◯ Todos os dias

◯ Algumas vezes por semana

◯ Uma vez por semana

◯ Uma vez por mês

◯ Nunca

B.2 Cultural Center survey

Q1. Durante a utilização do ActivitySpot, percebi claramente quais as acções ou funcionalidades que era possível executar.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q2. Verifiquei que as diferentes acções ou funcionalidades que executei estavam relacionadas entre si e faziam parte de um conjunto destinado a suportar os espectadores do CCVF.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q3. Verifiquei que executei funcionalidades que tiveram influência em outras funcionalidades executadas mais tarde.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q4. Foi possível compreender o efeito de todas as minhas interacções com o sistema.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q5. Foi possível, ao longo da utilização do ActivitySpot, ter conhecimento do estado da minha actividade (o que tinha feito, o que podia ainda fazer, etc.).

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q6. Não tive quaisquer dificuldades em perceber e utilizar o sistema através dos diferentes meios disponíveis (NOTA: responda apenas para os meios de interacção que tenha utilizado):

	Concordo totalmente	Concordo parcialmente	Discordo parcialmente	Discordo totalmente
SMS				
Cartão e porta-chaves de identificação				
Câmara do telemóvel				

Q7. O sistema ActivitySpot é adequado para ser utilizado nos espectáculos

do CCVF.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q8. O sistema ActivitySpot não é, para quem o utiliza, um factor de perturbação de uma normal ida ao CCVF.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q9. A utilização do sistema ActivitySpot foi frustrante.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q10. A utilização do sistema ActivitySpot tornou a minha ida ao CCVF mais interessante.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q11. Teria utilizado o sistema ActivitySpot espontaneamente, sem ter sido previamente convidado(a) a tal.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q12. Considero relevante a existência de sistemas como o ActivitySpot em outros espaços públicos.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q13. O sistema ActivitySpot forneceu-me informação personalizada, i.e., diferente daquela a que outros utilizadores tiveram acesso.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q14. O acesso a informação personalizada torna a utilização deste tipo de sistemas mais interessante.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q15. A utilização do ActivitySpot teria sido mais interessante se o sistema tivesse acesso a mais informação pessoal.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q16. O tempo disponível, em cada espectáculo, para a utilização do ActivitySpot é suficiente.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q17. As instruções de utilização fornecidas foram suficientes.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q18. O sistema teria sido mais fácil de utilizar se houvesse mais ajuda disponível.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q19. O sistema reagiu rapidamente às minhas interacções.

◯ Concordo totalmente

◯ Concordo parcialmente

◯ Discordo parcialmente

◯ Discordo totalmente

Q20. Sinto-me à vontade na utilização das seguintes tecnologias:

	Concordo totalmente	Concordo parcialmente	Discordo parcialmente	Discordo totalmente
SMS				
Cartões de identificação				
Ecrãs públicos				
Câmara do telemóvel				
Bluetooth				

B.3 Conference surveys

The conference surveys were mainly targeted at the experimental group, who used ActivitySpot. A short survey was also submitted to a control group, in order to assess whether ActivitySpot helped its users to better achieve their goals, compared to people who did not use ActivitySpot (control group). There were different survey formulations, depending on the role of the subject: one for conference participants in general, another for conference authors, and another for conference organizers. The following formulations were targeted at conference authors.

B.3.1 Experimental group

Q1. Foi possível perceber claramente o que o sistema ActivitySpot, na globalidade, pretendeu oferecer aos utentes do espaço da conferência.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q2. De entre as funcionalidades que o ActivitySpot possibilitou aos seus utilizadores, lembro-me das seguintes (sem consulta):

1. ____________________

2. ____________________

3. ____________________

Q3. Enfrentei dificuldades ao registar-me no ActivitySpot.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q4. O tempo despendido quando do registo no ActivitySpot é razoável.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q5. Na generalidade, a execução das funcionalidades oferecidas pelo ActivitySpot é complicada.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q6. Houve alguma funcionalidade que lhe tenha colocado dificuldades?

◯ Sim. Quais e porquê: ____________________

◯ Não

Q7. Utilizou o ActivitySpot baseando-se apenas nas instruções disponíveis (sem necessitar de qualquer ajuda humana)?

◯ Sim

◯ Não

Q8. O ActivitySpot teria sido mais fácil de utilizar se houvesse mais ajuda disponível.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q9. As diferentes interacções que efectuei estavam integradas entre si.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q10. As funcionalidades que executei tiveram influência em outras funcionalidades executadas mais tarde.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q11. O ActivitySpot respondeu às minhas interacções sempre da maneira esperada.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q12. Houve ambiguidade quanto ao destinatário das respostas do ActivitySpot.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q13. Ocorreu algum conflito com outros participantes na utilização concorrente dos meios de interacção disponibilizados?

◯ Sim

◯ Não

Q14. Ao longo da utilização do ActivitySpot, foi possível ir tendo conhecimento do estado da minha actividade.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q15. O esforço requerido pelo ActivitySpot permitiu a normal realização da minha actividade na conferência.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q16. A utilização do ActivitySpot distraiu-me da normal realização da minha actividade na conferência.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q17. A forma como as funcionalidades do ActivitySpot me foram oferecidas é coerente com a minha visão da actividade que costumo desenvolver numa conferência.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q18. O ActivitySpot forneceu-me informação personalizada, i.e., diferente daquela a que outros utilizadores tiveram acesso para as mesmas situações.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q19. A utilização do ActivitySpot teria sido mais interessante se o sistema tivesse acesso a mais informação pessoal.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q20. A utilização do ActivitySpot ajudou-me a realizar os objectivos da minha actividade.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q21. Numa escala de 1 (totalmente falhado) a 10 (totalmente realizado), o grau de realização do objectivo "conhecer trabalho nesta área científica", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	

Q22. Numa escala de 1 a 10, o grau de realização do objectivo "intercâmbio de conhecimento", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q23. Numa escala de 1 a 10, o grau de realização do objectivo "estabelecer contacto com outros investigadores", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q24. Numa escala de 1 a 10, o grau de realização do objectivo "recolher opiniões sobre o próprio trabalho", com ou sem ajuda do ActivitySpot, foi[1]:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q25. Caso não tivesse utilizado o ActivitySpot, os objectivos da minha actividade na conferência teriam sido atingidos com o mesmo sucesso.

○ Discordo totalmente

○ Discordo parcialmente

○ Concordo parcialmente

○ Concordo totalmente

Q26. A utilização de um sistema como o ActivitySpot em espaços públicos é preferível a uma única aplicação, instalada num telemóvel, que reúna as mesmas funcionalidades.

○ Discordo totalmente

○ Discordo parcialmente

○ Concordo parcialmente

[1]This question had a different formulation in the survey submitted to conference organizers or participants who were not authors.

◯ Concordo totalmente

Q27. A utilização de um sistema como o ActivitySpot em espaços públicos é preferível a uma única aplicação, instalada num quiosque interactivo, que reúna as mesmas funcionalidades.

◯ Discordo totalmente

◯ Discordo parcialmente

◯ Concordo parcialmente

◯ Concordo totalmente

Q28. Sinto-me à vontade na utilização das seguintes tecnologias:

	Discordo totalmente	Discordo parcialmente	Concordo parcialmente	Concordo totalmente
SMS				
Cartões de identificação				
Ecrãs públicos				
Câmara do telemóvel				
Bluetooth				

B.3.2 Control group

Q1. Numa escala de 1 (totalmente falhado) a 10 (totalmente realizado), o grau de realização do objectivo "conhecer trabalho nesta área científica", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q2. Numa escala de 1 a 10, o grau de realização do objectivo "intercâmbio de conhecimento", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q3. Numa escala de 1 a 10, o grau de realização do objectivo "estabelecer contacto com outros investigadores", com ou sem ajuda do ActivitySpot, foi:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

Q4. Numa escala de 1 a 10, o grau de realização do objectivo "recolher opiniões sobre o próprio trabalho", com ou sem ajuda do ActivitySpot, foi[2]:

Falhado	1	2	3	4	5	6	7	8	9	10	Realizado
	○	○	○	○	○	○	○	○	○	○	

[2]This question had a different formulation in the survey submitted to conference organizers or participants who were not authors.