
From Applications to Ubiquitous Instrumental Interaction

Clemens Nylandsted Klokrose

Department of Computer Science,
University of Aarhus
Åbogade 34, 8200 Århus N,
Denmark
clemens@daimi.au.dk

Michel Beaudouin-Lafon

LRI,
Univ. Paris-Sud
Bât 490 - Orsay, F-91405,
France
mbl@lri.fr

Abstract

This paper shows the limitations of the current application-centric approach to user interfaces when considering interaction in Ubicomp environments, and presents an alternative paradigm based on instrumental interaction. The paper then discusses the elements of a description language for ubiquitous instrumental interaction and outlines some challenges related to the sharing and distribution of instruments.

Keywords

HCI, ubiquitous computing, instrumental interaction, description languages

ACM Classification Keywords

H.5 Information interfaces and presentation

Copyright is held by the author/owner(s).

CHI 2007, April 28 – May 3, 2007, San Jose, USA

ACM 1-xxxxxx

Introduction

Ubiquitous interaction – interaction in pervasive, ubiquitous, tangible or ambient computing environments, including interaction with multiple, dynamic, and distributed interfaces – is an area where there currently is limited coherent theory to guide design. Unlike the desktop metaphor for office work, there is as yet no well-established metaphorical approach or conceptual model to guide the design of ubiquitous interaction. Ubiquitous interaction challenges the traditional assumptions of one device / one interface / one user in many ways: multiple users can interact with multiple devices through a variety of interfaces, including interfaces spanning multiples devices. The WIMP¹ paradigm and the desktop metaphor do not scale to these new situations.

In addition, the software tools used for creating interfaces are tightly bound to the platform hosting them and to the WIMP interaction style. These tools typically do not support the multiplicity, dynamism, heterogeneity and distribution that characterize Ubiquitous Interaction, making it particularly difficult to develop interfaces for Ubicomp environments.

¹ Windows, Icons, Menus and Pointing

An Instrumental Paradigm

One of the central goals in creating interfaces for Ubicomp environment is to support fluid interaction in distributed interfaces and interaction in dynamic configurations of interfaces distributed across stationary and mobile devices. Two major challenges that we want to address are: How to support the reuse and the quality of learning across different devices [4], and how to technically support the continuity and distribution of work across multiple devices.

We believe that one approach to address this problem is to *deconstruct* applications rather than simply create scaled-down versions of PC applications to run on, e.g., Personal Digital Assistants (PDAs). Scaling down applications leads to many problems: the tool has to be implemented on each device, an exact copy of functionality is hard to achieve, and the user has to learn to use the alternate implementation. Moreover the user must rely on the vendor to actually support the tools he needs.

The notion of *application* is indeed very artificial, with no direct equivalent in the real world. While applications can be seen as collections of tools dedicated to a certain task in the physical world, such as the architect's or the painter's tools, applications lack the dynamics of such collections. A painter can freely add or remove brushes from his collection, pass them around, etc., but a brush in a drawing application can rarely be removed and used in another context. Instead, applications typically have a predefined set of tools that is difficult or impossible to reconfigure to adapt to one's taste. This lack of flexibility limits the mobility, distribution and customizability of interfaces. It also typically results in large and complex

applications, built for general-purpose personal computers, with dozens and sometimes hundreds of tools to cover all possible needs. Such applications are not suitable for smaller devices, or devices with different kinds of inputs. This in turn creates the need for specific versions of the applications for different devices, which might be radically different across different platforms and technologies.

By contrast, the physical tools we use in the real world may have limited properties, but they can be put to unlimited uses and are usually easy to understand and adapt to one's need. For example, almost any surface, not just paper, affords to be written on with a pen. Collections of physical tools can assist in achieving complex goals, and they can be split or recomposed to address specific situations. For example, the architect can bring the drawing, a pencil and an eraser with him to a meeting. He can use the same pencil from the drawing table to write his grocery list, or he can use the pen with which he signed the check at the restaurant to draw a sketch on a paper napkin. In other words the tools are not restricted to a specific domain, they can be used across domains – and across people.

Letting the nature of the physical world's relationships between tools and objects be the ultimate goal would be naïve and overly idealistic. Nevertheless the kind of implicit relationships between us, our tools and the objects we interact with, which Gibson [5] calls *affordances*, can be approximated through the concept of *Instrumental Interaction* [2]. In this model, the coupling between tools and objects is not type-based, e.g. a given application for a given filetype, instead it is property-based, mapping one or more instruments to each object property or set of properties.

Grace is a graphics designer in a small advertising bureau, and is presenting a poster she has developed on her workstation for a client at the client's office. When seeing the poster on print the client asks if it would be possible to create flyers matching the design. Grace asks for a minute and while the client is watching she rearranges, trims and scales the poster into a flyer on her PDA using a subset of the tools, which she had used on the workstation and brought with her. The client is satisfied and Grace returns to her office workstation to give the finishing touches to the flyer.

Textbox 1. Interaction Scenario

Beaudouin-Lafon [2] introduced *Instrumental Interaction* to model WIMP and post-WIMP interfaces. The key idea is a conceptual separation between physical instruments (the input devices), logical instruments and domain objects. Logical instruments are software mediators or two-way transducers between the user and domain objects. The user acts on the instrument, which transforms the user's actions into commands that affect the relevant target domain objects and provide feedback to the instrument.

Inspired by this separation between instrument and domain object and by the notion that interaction is mediated by instruments, we suggest extending Instrumental Interaction to a general paradigm for ubiquitous and distributed interaction. We suggest an approach where functionality is implemented by instruments with limited properties, decoupled from the specific types of domain objects they operate on. The implicit relations between the properties of the object and the instruments operationalize, in a sense, Gibson's affordances. As a result, an instrument can operate on an object even if it has not been explicitly designed for it. For example, an object that has the property of being a 2D surface can be used by any instrument that operates on a 2D surface.

We further suggest to decouple the physical instruments – the input devices – from the logical instruments. This decoupling is necessary to support distribution of instruments or tools across devices. The logical instrument is not bound to a specific set of input devices but rather to any input device that has the properties described by the logical instrument, such as providing 2D input vs. text input.

To give an example of the kind of interaction we imagine, consider the simple scenario in textbox 1 to the left. The scenario illustrates the fluidity of interaction when using instrumental interaction in an Ubicomp environment.

The decomposition of user interfaces into simpler commands is not new. Jacob [6] describes, in one of the initial attempts at creating UIDLs for modern direct manipulation interfaces, interaction in direct manipulation systems as composed by a set of simple dialogues. Beaudouin-Lafon [2] emphasizes mediation rather than direct engagement with the system, inspired by activity theory [3]. A mediated relationship between the user and the objects of interest, in contrast to dialog-based interaction, is particularly attractive for ubiquitous interaction since the notion of system or computer – the other participant in the dialogue – becomes blurry and changing. The notion of mediation is also useful when dealing with detached instruments that act on different objects in different contexts.

Descriptive Challenges in Realization

The dynamic and distributed nature of ubiquitous interaction requires both descriptions and state of components to be easily shared among devices. Hence high-level description languages seem to be an appropriate approach.

We now give an overview of the descriptive challenges in implementing the ubiquitous instrumental interaction that we have outlined. The challenges can roughly be divided into *object, need/capability mapping* and *state/behavior descriptions*.

The components that need descriptions are the objects of interest, the instruments, the devices (resources and i/o), and the representations of the objects or views. First, a language is needed to describe the object model (general or domain specific). This is a fairly common and well-understood problem.

Second, we need to describe (logical) instruments: their input needs, e.g., text input vs. 2D coordinates, and their capabilities for manipulating certain properties of the domain objects. The needs of the instruments will have to match the capabilities of the input devices, e.g. a mouse for 2D coordinates. The behavior of the instrument could be described using state machines as in Jacob's dialogue descriptions [6] or ala SwingStates [1]. An interesting challenge is to support distributed interactions, e.g., a paintbrush instrument with the canvas on one device and the palette on another, which requires sharing or distributing the state machines among the interested parties, in a dynamic way.

Third, we need to describe views, taking into account the possible migration of objects among surfaces managed by different devices. Finally, on top of the description of the components, a language is required to describe the various mapping between instruments and input devices, between instruments and objects and between views and output devices.

Conclusion

This article has given a brief overview of ubiquitous instrumental interaction and has outlined the elements of a description language. We have illustrated some non-trivial challenges arising from migrating and sharing instruments across devices. A first prototype is

being developed in the first author's lab in order to experiment with the concept of instrumental interaction in an Ubicomp environment. We will also use this prototype to assess the requirements for tools and languages to develop such interfaces.

Acknowledgements

We thank Susanne Bødker, Olav Bertelsen and Pär-Ola Zander for valuable input and discussions and give acknowledgements to Rasmus Berlin for being the critical programmer.

Citations

- [1] Appert, C. and Beaudouin-Lafon, M. 2006. SwingStates: Adding State Machines to the Swing Toolkit. In Proceedings of ACM symposium on User Interface Software and Technology (Montreux, Suisse, October 16 - 18, 2006). UIST'06. ACM Press, New York, NY, April 2006, pages 319-322.
- [2] Beaudouin-Lafon, M. (2000). Instrumental interaction: an interaction model for designing post-WIMP user interfaces. CHI 2000. ACM Press.
- [3] Bertelsen, O. W. and Bødker, S. (2003). *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*, chapter Activity Theory. Morgan Kaufman Publishers.
- [4] Brodersen, C., Bødker, S. and Klokmoose, C. N. (2007). Quality of Learning in Ubiquitous Interaction. European Conference on Cognitive Ergonomics 2007 (ECCE 2007). Covent Garden, London, UK, 28-31 August, 2007
- [5] Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, New Jersey, USA.
- [6] Jacob, R. J. (1986). A specification language for direct-manipulation user interfaces. ACM Trans. Graph. 5, 4 (Oct. 1986), 283-317.