

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

REVIEW ON COMPONENT MODELING APPROACHES FOR CONTEXT-AWARE APPLICATIONS AND INPUT DEVICES USED BY MOBILE DEVICES.

Diane Lingrand, Michel Riveill

Projet RAINBOW

Rapport de recherche
ISRN I3S/RR-2006-12-FR

Avril 2006

RÉSUMÉ :

Dans ce rapport, nous nous intéressons aux applications sur terminaux mobiles pour lesquels les interactions entre utilisateurs et équipements informatiques doivent être repensés. Effectivement, depuis quelques années, l'informatique ubiquitaire et en particulier l'informatique contextuelle, cherchent à apporter de nouvelles solutions.

Lors de l'utilisation d'équipements mobiles le contexte d'utilisation est encore plus variables qu'avec des terminaux fixes. Dans un tel environnement, les applications logicielles doivent évoluer dynamiquement. La programmation par composants est un paradigme de programmation facilitant la création de tels logiciels.

Nous allons tout d'abord définir précisément le contexte, les applications contextuelles ainsi que les interactions par périphériques d'entrée. Ensuite, nous présentons les différentes modélisations existantes, basées sur une approche par composants. Nous chercherons à étudier les différences et similarités entre la gestion du contexte et des périphériques d'entrée afin de conduire à une architecture commune.

MOTS CLÉS :

informatique contextuelle, périphériques d'entrée

ABSTRACT:

This paper deal with software applications on mobiles devices. One of the major challenges of these equipment concerns the interaction with these devices. Traditional interaction using mouse and keyboard as input devices are no more sufficient: they may be not available (how to type on a keyboard while walking) and interactions with an application may not be the main focus of the user (interacting with a computer is not the main task of a taxi driver). Different input devices taxonomies are proposed by authors. Since several years, ubiquitous computing and pervasive computing has emerged and, in particular, context-aware computing.

Using mobile devices, the context is perpetually evolving, more than with standard workstation. In such environment, software must modify its behavior dynamically. An emerging way of programming an adaptive software is component programming.

After defining the notion of context and input devices, we will review existing modeling approaches, especially those that consider component modeling. We aim at determining the similarities and differences between context and input devices in order to propose a common component model architecture that will help building such intelligent applications.

KEY WORDS :

context-aware computing, input devices interactions

Review on component modeling approaches for context-aware applications and input devices used by mobile devices.

Diane Lingrand, Michel Riveill

RAINBOW team

I3S (Univ. Nice - Sophia Antipolis/CNRS) UMR 6070

B.P. 121 – 06903 Sophia Antipolis - FRANCE

{Diane.Lingrand, Michel.Riveill}@unice.fr

Abstract

This paper deal with software applications on mobiles devices. One of the major challenges of these equipment concerns the interaction with these devices [Sta01]. Traditional interaction using mouse and keyboard as input devices are no more sufficient: they may be not available (how to type on a keyboard while walking) and interactions with an application may not be the main focus of the user (interacting with a computer is not the main task of a taxi driver). Different input devices taxonomies are proposed by authors. Since several years, ubiquitous computing and pervasive computing [Wei91] has emerged and, in particular, context-aware computing.

Using mobile devices, the context is perpetually evolving, more than with standard workstation. In such environment, software must modify its behavior dynamically. An emerging way of programming an adaptive software is component programming.

After defining the notion of context and input devices, we will review existing modeling approaches, especially those that consider component modeling. We aim at determining the similarities and differences between context and input devices in order to propose a common component model architecture that will help building such intelligent applications.

1 Introduction

This paper deals with software applications on mobiles devices. One of the major challenges of developing such applications concerns the interaction with these devices [Sta01]. Traditional interaction using mouse and keyboard as input devices are no more sufficient: they may be not available (how to type on a keyboard while walking) and interactions with an application may not be the main focus of the user (interacting with a computer is obviously not the main task of a taxi driver while driving).

Adapting the software to the situations, to the task to be performed and to the user could decrease the need of explicit interactions between the user and the application. This is part of ubiquitous computing approach and is context-aware computing aim: building applications that are aware of the context.

In order to build software that can dynamically adapt to a context that is constantly evolving, the component modeling approach is the most appropriate. This approach has emerged since several years, in the continuity of modular coding and objects programming. After focusing on standalone software running on one workstation, we are now in a world of complex software applications that are distributed and that run on different hardware platforms. The complete rewriting of an application is no more thinkable. Portions of code must be reused. Software must dynamically adapt to evolving situations.

However, the notion of component is not clearly defined. We can say that it has inputs, outputs that are computed from the inputs. An interesting property is that the composition of two components is also a component. But composition is also not really clearly defined.

The compositional adaptation is defined in [MSKC04] as it “enables software to modify its structure and behavior dynamically in response to changes in its execution environment”.

The separation of concerns between business components and technical components has been studied by several authors and is now clearly well established in the component programming community.

However, the software part that is concerned by interactions needs to be modeled by components in order to take into account the plasticity of interfaces. Very little works has been done considering this GUI components approach, except [PDF03].

It is also necessary that the context acquisition and context interpretation itself dynamically adapt to evolving sensors, needs or tasks ... which means that the context processing need to be context-aware. For this matter, a component modeling approach for the context seems to be well adapted.

There are two different aspects concerning, context and composition: (i) the composition of context components for context acquisition and context processing and (ii) the adaptation of components to the evolving context. However, the availability of a sensor may also be part of the context: context acquisition is then depending on the context ! Therefore, we will clarify the notion of context later in the paper.

The focus of this paper is to review existing component approaches for input devices and for context. The question is: what can we learn from context-aware modeling for input devices management and vice versa ? We aim at determining the similarities and differences between context and input devices in order to propose a common component model architecture that we help building such intelligent applications.

We will first review component approach for input devices. As we will see, they are not widely studied. Then, we will focus on context component architectures that have been more widely studied.

2 Input devices and taxonomies

Input devices are devices used by users to enter commands to computers such as mouse and keyboard. There exist a large variety of input devices from keyboards to

3D mouse, speech recognition, handwriting recognition, video analysis and so on.

Input devices taxonomies Since several years, several authors have proposed input devices taxonomies in order to classify these devices. Foley et Wallace [FW74] classify the input devices in four families: grab, localization, button and value. In [FWC84], authors propose to associate input devices to actions to be performed.

In other hand, Buxton [Bux83] classify input devices on a bi-dimensional graph depending on the degrees of freedom and measured data (motion, pressure, ...) and the type of touch with the user. Card and colleagues [CMR91] have extended this taxonomy by distinguishing the absolute values from relative values and in separating translation components from rotation components. This classification is more precise than previous ones but does not allow an enough precise equivalence between devices or to determine the more appropriate device for a special task or situation.

Jacob et Sibert [JS92] began to observe that the missing dimension concerns the cognitive perception. Their taxonomy makes the distinction between integral devices (considering all degrees of freedom as a whole) and separable devices (consider each degree of freedom individually). This classification has been used by, among others, [HTP⁺97, BBKF97].

Beside these almost theoretical different taxonomies, we also find experimental comparison based on measures such as the Fitts law [Fit54] or the steering law [AZ97], as for example in [HTP⁺97].

Mackinlay and colleagues approach The first work that present input devices closer to the component approach is presented in [MCR90]. They observed that "neither taxonomy deals with the combination of individual input devices into complex input controls." which consist of a premise of component composition.

They represent an input device by a sextuplet : $\langle M, In, S, R, Out, W \rangle$ where M is the manipulation operator (e.g. z-axis rotation for a rotary plot), In the input domain (e.g. values between 0° and 270°), S the current state (e.g. θ), R the resolution function, Out the output domain and W works or rules (e.g. what to do if a joystick is released).

The authors propose three composition operators for Input Devices: connection composition, layout composition and merge composition. The connection composition is defined by;

$$\text{Connect} = \langle \text{Input Domain, Resolution Function, Output Domain} \rangle$$

and connects the output of a component to the input of another component. However, the use of the resolution function has been restricted to constant scale factor and clipping. More complex resolution functions could be studied in order to have more complex interactions.

The layout composition consists of a space repartition of devices (e.g. horizontally aligned). The merge composition concern different input devices merged in one (e.g. two 1D sensors merged in a 2D sensor): the merge composition of two input devices is still an input devices.

Phidgets In [Res93], authors review methods of bricks composition, from composition for kids based on LEGO and LOGO to Electronic Bricks and to Programmable Bricks. One step further, the phidgets [GF01] (physical widgets) are programmable components representing physical devices. Authors argue that “just as widgets make GUIs easy to develop, so could phidgets make the new generation of physical user interfaces easy to develop”. It is easy to build new devices by physical assembly of individual phidgets.

VRPN The Virtual-Reality Peripheral Network (VRPN) [THS⁺01] is a library for transparent network interface between application software running on a virtual-reality dedicated computer and computer there physical devices are connected. VRPN provides an abstraction layer that makes all devices of the same base class look the same. VRPN is an interesting approach but does not provide higher level mechanisms for composition.

Graphical toolboxes In other hand, the ICON (Input Configurator) [DF01, DF04] approach is a toolbox for easy creating Post-WIMP¹ input devices. They propose a new input model based on input configurations. The ICON input toolkit is composed by three different types of component: *system devices* (input devices) that can be linked to *library devices* (transformations providing higher level informations) and *application devices* that allow users to build interactive applications and to configure their inputs in order to use alternative input device and techniques taking into account the use of impoverished or enriched input (see figure ??). More recently, the WCOMP [CFWJG⁺03] approach consider input devices as JavaBean component. Both approaches allow the programmer to graphically choose in the composition schema the component management he wants for the input devices. These two approaches provide mechanisms to allow the programmer to compose components.

Conclusion on input devices representation The Mackinlay approach is a good basis for an input device component. However, it is not enough expressive for composition. The composition proposed by the Phidgets approach is interesting but not sufficient for devices of higher level. The ICON approach is the closest to what we are looking for because all devices existing today are considered and the component architecture separating the acquisition (devices) from analysis and presentation to the application. However, it is at the user to program the adaptation.

3 Context-aware applications

We will now focus our interest on context-aware applications. These applications use informations from their environment such as it were a kind of input device. This field of research is really younger than the field of input interactions and is subject of a lot of different ideas on components based architecture developed by different teams.

¹WIMP : Windows Icons Menu Pointers

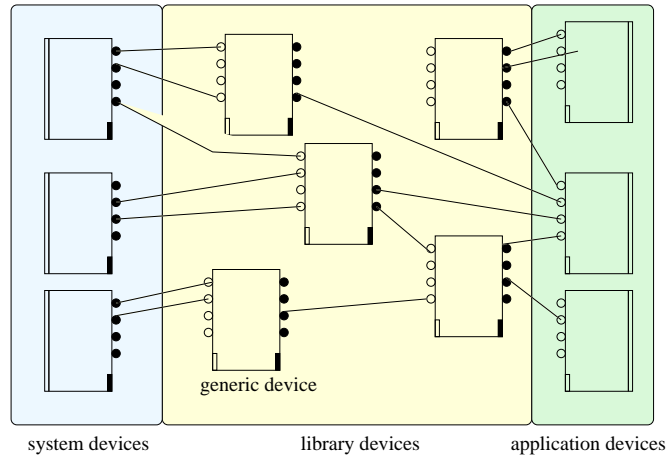


Figure 1: The ICON components architecture

What is context ? A definition that is general and well cited by authors in the context-aware computing community is the definition by Dey [Dey01]: "Any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves".

Several authors [CK00, CR02] have made a classification of the different types of context in several families: **environmental context** such as nearby people or objects, luminosity, temperature, noise or weather, **user context** such as user location, personal interest, preferences, activity, emotion, muscle contraction, blood pressure or handicap, **computer context** such as URL, nearby displays, network bandwidth, memory available and **time context** such as history of actions, history of locations or current time and date.

These informations have very different natures. Some are easily measurable using sensors such as the temperature measured by a thermometer and can be compared each others. Some are more difficult to measure such as emotion or weather because of the subjectivity of their nature. They need to be quantified with respect to the usage that will be done by applications. Some other information such as preferences should be entered by users or inferred from other informations on the user. For example, a user may enter his preferences about what lamp will be switch on when he enter a room or it can be inferred observing that each time this user enter this room, he switches some lamps on and some others off [DHB⁺04]. As another example, a positioning system such as the GPS gives three coordinates with different accuracy but, knowing a local map, information on the city, road, buildings may also be part of the context : different levels of abstraction are needed.

Therefor, some authors differentiate the **sensed context** from other type of context informations. In [GS01], authors defined sensed context as referring to that part of context that is accessible via sensors. They define the sensed context as

“properties that characterize a phenomenon, are sensed and that are potentially relevant to the tasks supported by an application and/or the means by which those tasks are performed.”

Henricksen [HI06] consider sensed, static, user-supplied (profiled) and derived information as the most useful. She also mention that since context comes from a wide variety of sources, the quality and persistence of context information is really heterogeneous. These four categories of context present distinct characteristics.

Context information may be provided by sensors that have different quality attributes such as accuracy, robustness, precision. They also could fail. Context information may also be provided by users. In this case, information is likely to be correct but may be not up-to-date if the user neglected to update them. Henricksen [HI04] classified the properties in four classes: unknown, ambiguous (two sensors give different responses), imprecise and erroneous. The authors of [HI04] precise the persistence, quality issues and sources of inaccuracy for the four types of context: sensed, static, profiled and derived.

Then, context information is not only the information itself but also needs meta-information such as accuracy, dynamic, and also cost function in order to compare different context each other [LLT05].

What are context-aware applications ? We need to distinguish parameterized applications, context-aware applications and context-dependent applications. Parameterized applications used informations as parameters without adaptation. Context-dependent applications cannot run without context information but may have adaptation to fluctuation of context (e.g. a GPS on board of a car). Context-aware applications are enhance by the presence of context information but can still run without information (e.g. a context-aware mobile phone will automatically switch the ring off entering a meeting and augment the level of ring in a noisy room in absence of a meeting but will still react as a usual mobile phone in absence of informations).

In [DSA01], three type of context-aware functions are defined:

- presenting information and services
- automatically executing a service
- attaching context information for later retrieval

In the following, we focus on the modelisation of context in components. Our need concern the adaptation of the system to the addition of a context component, the deletion of a context component and the replacement of one component by another.

Replacing, for example, a GPS receiver by another is a kind of driver problem. But replacing a localization system using a GPS by an indoor positioning system when the user enter a building induce new problem: the resolution is different, the precision is different and the information we will use may be different. Making measurements [LLT05] will need to have a semantic continuity in case of sensor missing or new sensor connecting.

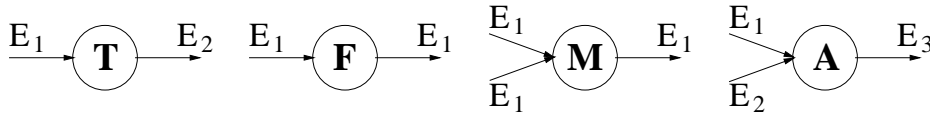


Figure 2: The four types of operators: Transformer (T), Filter (F), Merger (M) and Aggregator (A).

4 Component models of the context

4.1 The Context Toolkit and further approaches

The Context Toolkit The Context Toolkit (CTK) [SDA99, DA00, DSA01] provides applications with access to context information from their environment using software components that have distinct functions. Context widgets are responsible of the acquisition of context information. Four other components that are responsible of context-specific operations: interpreters, aggregators, services and discoverers. Interpreters produce higher-level context information from informations given by context widgets. Aggregators product an easy access to all the context informations related to a given entity. Then, services execute behaviors on the environment related to the context obtained and discoverers allow applications to know the context informations available in order to take advantages of them.

The CTK aims at hiding the complexity of sensors and data acquisition from sensors. The context information is abstracted to higher level in order to separate the acquisition concern to how context information is used. This toolkit is make from components that are reusable and customizable.

The Context Toolkit can be downloaded at contexttoolkit.sourceforge.net [ND03] and has been used by several people. An interesting experiment if the one of [BE02]: they have used the Quake III Arena which is composed by a game engine (closed) and control on objects behavior with the environment (opened). They have merged Quake III Arena and Context Toolkit to make QuakeSim in order to have an experimental platform for their context-aware applications.

However, it is not really clear how to merge informations from different sensors, how to maintain different levels of abstractions and how to use informations about sensors such as resolution, accuracy that are accessible.

The Solar System The Solar System [CK02] is a graph based approach following [DSA01] that proposes four types of operators (see figure 2: Transformer, Filter, Merger and Aggregator that can be combined. The differentiation of the operators is interesting but no information is given on how to compose these operators.

4.2 The Context-Handling Component and The Contextor

The Context-Handling Component The Context-Handling Component [GS01] is responsible of transformations on context data using three flows of informations:

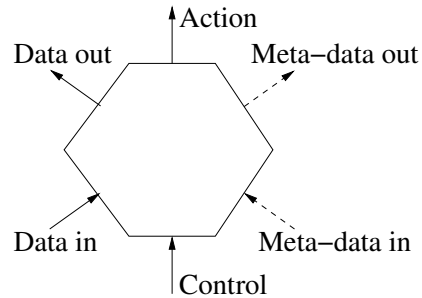
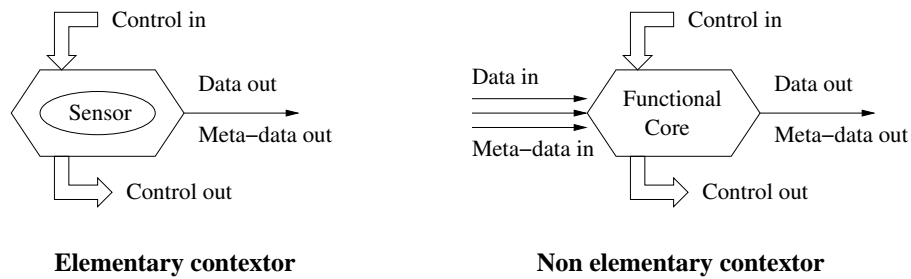


Figure 3: The Context Handling Component and its three flows: data, meta-data and control.



Elementary contextor

Non elementary contextor

Figure 4: The two different types of contextor.

data, meta-data (informations on data) and control (see figure 3). Each component can connect its outputs to the inputs of another component.

The Contextor The Contextor [CR02, RC04] is a computational model for context that is an enhancement of the context-handling component (see figure 4). Contrarily to the context-handling component, the contextor uses the data and meta-data in the same channel in order to insure synchronism.

It is a “software abstraction that models a relation between variables of an Observed System Context” and “returns the value of a variable that belongs to that context”. The authors present different contextor types: the elementary contextor that does not have data and meta-data entries because this contextor represents sensors and the non elementary contextors: history contextor, threshold contextor, translation contextor, fusion contextor and abstraction contextor (see figure 4).

Authors define a situation as a set of values observed and the context as a composition of multiple situations over a period of time. The composition of contextors is defined as “a kind of history function that involves the emergence of new relations and peripheral state variables as well as the destruction of old ones” but without explained how it is performed.

They explicit two types of composition, the “Data Channels Connection” that

is dynamically done at the execution and the “Encapsulation” that is decided by the software writer.

4.3 The Strathclyde Context Infrastructure (SCI)

In [TTN03], authors imagine a context entity as a software component. They consider the problem of discovering and composing appropriate context entities to be a special case of the more general problem of discovering and composing components in software engineering. They also abstract the concrete context entities and configurations to basic context elements and context operators, in order to build higher-level contexts.

The Strathclyde Context Infrastructure (SCI) [GSR⁺03, TTN03] is made from two distinct layers. The upper layer of the infrastructure (the SCINET) is a network overlay of partially connected nodes or ranges which represent a bounded physical or logical area. The lower layer of the infrastructure concerns the contents of each node made from different entities: People, Software, Places, Devices and Artifacts. This layer is responsible for producing, managing and using contextual information. In each range, a single Context Server manages three types of components: Context Entities, software component for representing an entity within the infrastructure, Context Utilities, set of specialist services that help the Context Server in the management of a Range and Context Aware Applications that have the ability to pull or be pushed contextual information to or from the infrastructure.

They address several open issues such as the dynamic composition of context entities, control over the quality and structure of context composition, and adaptivity to environmental changes.

4.4 The Sentient object model (CORTEX project)

The sentient object programming model [BC04, WFB⁺04] is a component-oriented reflective middleware based on OpenCOM aiming at sensor fusion, context extraction and reasoning. It is composed by three different categories of software entities: sensors, sentient objects and actuators.

Sentient objects are intelligent software components that are able to act autonomously based on acquisition information and to cooperate with each others.

The sentient object presents different characteristics: sentience (the ability to perceive the state of the environment via sensors), autonomy (the ability to operate independently of human control in a decentralized manner) and pro activeness (the ability to act in anticipation of future goals or problems).

The communication is event-based: sensors produce software events that are consumed by sentient objects who produce in response software events that are consumed by actuators (see figure 5). Actuators react by attempting to change the state of the real world via some hardware device.

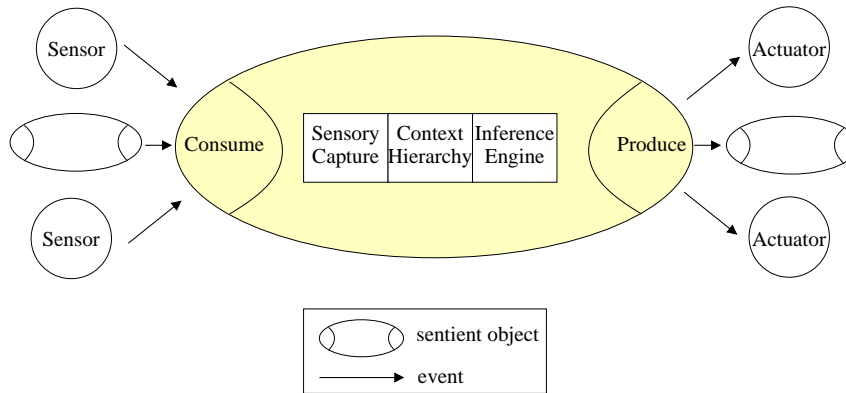


Figure 5: The sentient object programming model

4.5 The iQueue framework

This framework [CPWY02] aims at making software creation easier by facilitating the mechanics of composition. In this event-based framework, programmers choose events which are of interest for the context, specify fusion services using Bayesian networks and specify rules using an embedded inference engine. This should allow the programmer to focus only on the semantic of context use and composition.

4.6 Missing requirements for context-aware applications

Some of the previously presented architectures allow the programmer to easily compose dynamically context components: addition of a new sensor, deletion of a sensor or replacement of a sensor by another. However, the automatic adaptation of an application to missing context informations or new context informations is not completely addressed.

In order to evaluate and to compare different contextual situations one from another, we need a measure of a context. In [LLT05], we have introduced a notion of cost function. But cost composition can be done only on data that represent the same physical quantity. As for example, a cost in term of time, in term of money. In these cases, an addition of all the elementary costs will be done. Then, for one sensor, we need different elementary cost functions depending on the usage that will be done in the applications. Could it be foreseen? Accuracy on the data must also be taken into account. We then need to imagine more complex rules defined by the user, such as a user profile.

5 Discussion

Sensed context informations and user input interactions are both acquired using different sensors. Depending of the situation, some of them have a great confidence (e.g. push button or information entered by a user in a comfortable position) while

others are sensitive to noise (e.g. speech recognition, space localization or push button in transportation).

Input devices are generally intentionally used by the user in order to interact with an application while context acquisition is passive. The borderline is tiny: if a user knows how the system reacts to context modifications, he could use this information in order to modify the system: users always want to divert systems !

However, input devices are usually used for one application at a time. Since context-aware use needs to be transparent to the user, one context information can be used simultaneously by different applications, making it more difficult to be diverted by the user. The part of the system that is responsible of context management is independent of applications as it is for input devices since a long time!

The previously presented architectures for context or component models for input devices (4) always separate the acquisition step that is common to all applications from the interpretation one that depends on the use of the informations that will be done by the application.

The future challenges concern essentially the integration of input devices and context interpretation components at the heart of the application architecture in order to adapt them during application reconfiguration. The study briefly presented in this paper, but concerning more largely software architectures, shows that different software architectures are interesting in solving the reconfiguration problem. As pointed out by [BDR06], there is a need to precise the semantic of composition in order to dynamically associate software components that constitute the business part of the application and to allow the application to interact with its environment.

References

- [AZ97] Johnny Accot and Shumin Zhai. Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. In *Conference on Human Factors and Computing Systems (CHI)*, pages 295–302. ACM, 1997.
- [BBKF97] Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, and George Fitzmaurice. The Rockin' Mouse: Integral 3D Manipulation on a Plane. In *Conference on Human Factors and Computing Systems (CHI)*, pages 311–318. ACM, 1997.
- [BC04] Gregory Biegel and Vinny Cahill. A Framework for Developing Mobile, Context-aware Applications. In *IEEE Conference on Pervasive Computing and Communications (PerCom)*, Orlando (Florida, USA), March 2004.
- [BDR06] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A Survey of Context-Aware Systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2006.
- [BE02] Markus Bylund and Fredrik Espinoza. Testing and Demonstrating Context-Aware Services with Quake III Arena. *Communications of the ACM*, 45(1):46–48, January 2002.
- [Bux83] William Buxton. Lexical and Pragmatic Considerations of Input Structures. *ACM SIGGRAPH Computer Graphics*, 17(1):31–37, 1983.
- [CFWJG⁺03] Daniel Cheung Foo Woo, Gabriel Joulie, Florent Grillon, Jérôme Fuchet, and Jean-Yves Tigli. Wcomp: Rapid Application Development Toolkit for Wearable Computer Based on Java. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4198–4203, Washington DC (USA), October 2003.

- [CK00] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth Coll, November 2000.
- [CK02] Guanling Chen and David Kotz. Context Aggregation and Dissemination in Ubiquitous Computing Systems. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 105–116, Callicoon (NY, USA), June 2002. IEEE Computer Society Press.
- [CMR91] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A Morphological Analysis of the Design Space of Input Devices. *ACM Transactions on Information Systems*, 9(2):99–122, April 1991.
- [CPWY02] Norman H. Cohen, Apratim Purakayastha, Luke Wong, and Danny L. Yeh. iQueue: A Pervasive Data Composition Framework. In *International Conference on Mobile Data Management (MDM)*, page 146, Washington (DC, USA), 2002. IEEE Computer Society.
- [CR02] Joëlle Coutaz and Gaëtan Rey. Foundations for a Theory of Contextors. In *International Conference on Computer-Aided Design of User Interfaces (CADUI)*, pages 283–302, Valenciennes (France), May 2002. ACM Press.
- [DA00] Anind K. Dey and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Aware Applications. In *Workshop on Software Engineering for Wearable and Pervasive Computing (SEWPC)*, Limerick (Ireland), June 2000.
- [Dey01] Anind K. Dey. Understanding and Using Context. *Pattern Recognition Letters (PRL)*, 5(1):4–7, 2001.
- [DF01] Pierre Dragicevic and Jean-Daniel Fekete. Input device selection and interaction configuration with ICON. In *International Conference IHM-HCI*, pages 543–548, Lille (France), 2001. British HCI Group and AFIHM.
- [DF04] Pierre Dragicevic and Jean-Daniel Fekete. The Input Configurator Toolkit: Towards High Input Adaptability in Interactive Applications. In *Working Conference on Advanced Visual Interfaces (AVI)*, pages 244–247, Gallipoli (Italy), May 2004. ACM Press.
- [DHB⁺04] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: Programming by demonstration of context-aware applications. In *ACM Conference on Human Factors in Computing Systems (CHI)*, CHI Letters 6(1), April 2004.
- [DSA01] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI)*, 16(2-4):97–166, 2001.
- [Fit54] Paul Fitts. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, 46:381–391, 1954.
- [FW74] James D. Foley and Victor L. Wallace. The Art of Natural Graphics Man-Machine Conversation. *Proceedings of the IEEE*, 62(4):462–470, April 1974.
- [FWC84] James D. Foley, Victor L. Wallace, and Peggy Chan. The Human Factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics and Applications*, 4(11), November 1984.
- [GF01] Saul Greenberg and Chester Fitchett. Easy development of physical interfaces through physical widgets. In *Symposium on User Interface Software and Technology (UIST)*, pages 209–218, Orlando (Florida), November 2001. ACM Press.
- [GS01] Philip D. Gray and Daniel Salber. Modelling and Using Sensed Context Information in the Design of Interactive Applications. In *8th IFIP International Conference on Engineering for Human-Computer Interaction (EHCI)*, volume LNCS 2254, pages 317–336, Toronto (Canada), May 2001. Springer-Verlag.

- [GSR⁺03] Richard Glassey, Graeme Stevenson, Matthew Richmond, Paddy Nixon, Sotirios Terzis, Feng Wang, and Ian Ferguson. Towards a Middleware for Generalised Context Management. In *Workshop on Middleware for Pervasive and Ad Hoc Computing, Middleware (MPAC)*, June 2003.
- [HI04] Karen Henriksen and Jadwiga Indulska. Modelling and using imperfect context information. In *Workshop on Context Modeling and Reasoning (CoMoRea)*, pages 33–37. EEE Computer Society, March 2004.
- [HI06] Karen Henriksen and Jadwiga Indulska. Developing context-aware pervasive computing applications: Models and approach. *Journal of Pervasive and Mobile Computing (PMC)*, 2(1):37–64, February 2006.
- [HTP⁺97] Ken Hinckley, Joe Tullio, Randy F. Pausch, Dennis Proffitt, and Neal F. Kassel. Usability Analysis of 3D Rotation Techniques. In *Symposium on User Interface Software and Technology (UIST)*, pages 1–10. ACM, 1997.
- [JS92] Robert JK Jacob and Linda E. Sibert. The Perceptual Structure of Multidimensional Input Devices. In *Conference on Human Factors and Computing Systems (CHI)*, pages 211–218. ACM, 1992.
- [Lin06] Diane Lingrand. Rainbow bibliography tool - Outil de gestion bibliographique. Technical Report I3S/RT-2006-09-FR, I3S, Sophia Antipolis (France), March 2006. <http://diane.polytech.unice.fr/biblio/>.
- [LLT05] Diane Lingrand, Stéphane Lavirotte, and Jean-Yves Tigli. Selection using non symmetric context areas. In *Workshop on Context-Aware Mobile Systems (CAMS)*, volume LNCS 3762, pages 225–228, Agia Napa (Cyprus), October 2005. OnTheMove Federated Conferences (OTM'05), Springer.
- [MCR90] Jock D. Mackinlay, Stuart K. Card, and George G. Robertson. A Semantic Analysis of the Design Space of Input Devices. *Human-Computer Interaction (HCI)*, 5:145–190, 1990.
- [MSKC04] Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H. C. Cheng. Composing Adaptive Software. *IEEE Computer*, 37(7):56–64, July 2004.
- [ND03] Alan Newberger and Anind K. Dey. Designer Support for Context Monitoring and Control. Technical Report IRB-TR-03-017, Intel Research Berkeley, June 2003.
- [PDF03] Anne-Marie Pinna-Déry and Jérémy Fierstone. Component model and programming: a first step to manage Human Computer Interaction Adaptation. In *5th International Symposium on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI)*, volume LNCS 2795, pages 456–460, Udine (Italy), September 2003. L. Chittaro (Ed.), Springer Verlag.
- [RC04] Gaëtan Rey and Joëlle Coutaz. The Contextor Infrastructure for Context-Aware Computing. In *Workshop on Component-Oriented Approaches to Context-Aware Computing (COACAC, ECOOP)*, volume LNCS 3344, Oslo (Norway), June 2004. Springer Verlag.
- [Res93] Mitchel Resnick. Behavior Construction Kits. *Communications of the ACM*, 36(7):64–71, July 1993.
- [SDA99] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Conference on Human Factors in Computing Systems (CHI)*, pages 434–441, Pittsburgh, May 1999. ACM Press.
- [Sta01] Thad Starner. The Challenges of Wearable Computing. *IEEE Micro*, pages 44–67, July 2001.
- [THS⁺01] Russell M. Taylor, Tom Hudson, Adam Seeger, Hans Weber, Jeff Juliano, and Aron Helsen. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. In *ACM Symposium on Virtual Reality Software & Technology (VRST)*, Banff (Canada), November 2001. ACM, SIGGRAPH, and SIG-CHI, ACM Press.

- [TTN03] Graham Thomson, Sotirios Terzis, and Paddy Nixon. Towards Dynamic Context Discovery and Composition. In *UK-Ubinet Workshop*, London (England), 2003. Imperial College.
- [Wei91] Mark Weiser. The Computer for the Twenty-First Century. *Scientific American*, pages 94–104, September 1991.
- [WFB⁺04] Maomao Wu, Adrian Friday, Gordon Blair, Sivaharan Thirunavukkarasu, Paul Okanda, Hector Duran Limon, Carl-Fredrik Sorensen, Gregory Biegel, and René Meier. Novel Component Middleware for Building Dependable Sentient Computing Applications. In *Workshop on Component oriented approaches to Context-aware computing (ECOOP)*, Oslo (Norway), June 2004.

This bibliography has been generated using the Rainbow Bibliography Tool [Lin06].