Towards a Better Understanding of Context and Context-Awareness

Anind K. Dey and Gregory D. Abowd

Graphics, Visualization and Usability Center and College of Computing,
Georgia Institute of Technology, Atlanta, GA, USA 30332-0280
{anind, abowd}@cc.gatech.edu

Abstract. The use of context is important in interactive applications. It is particularly important for applications where the user's context is changing rapidly, such as in both handheld and ubiquitous computing. In order to better understand how we can use context and facilitate the building of context-aware applications, we need to more fully understand what constitutes a context-aware application and what context is. Towards this goal, we have surveyed existing work in context-aware computing. In this paper, we provide an overview of the results of this survey and, in particular, definitions and categories of context and context-aware. We conclude with recommendations for how this better understanding of context inform a framework for the development of context-aware applications.

1 Introduction

Humans are quite successful at conveying ideas to each other and reacting appropriately. This is due to many factors: the richness of the language they share, the common understanding of how the world works, and an implicit understanding of everyday situations. When humans talk with humans, they are able to use implicit situational information, or *context*, to increase the conversational bandwidth. Unfortunately, this ability to convey ideas does not transfer well to humans interacting with computers. In traditional interactive computing, users have an impoverished mechanism for providing input to computers. Consequently, computers are not currently enabled to take full advantage of the context of the human-computer dialogue. By improving the computer's access to context, we increase the richness of communication in human-computer interaction and make it possible to produce more useful computational services.

In order to use context effectively, we must understand both what context is and how it can be used. An understanding of context will enable application designers to choose what context to use in their applications. An understanding of how context can be used will help application designers determine what context-aware behaviors to support in their applications.

How do we as applications developers provide the context to the computers, or make those applications aware and responsive to the full context of human-computer interaction? We could require users explicitly to express all information relevant to a given situation. However, the goal of context-aware computing should be to make interacting with computers easier. Forcing users consciously to increase the amount of information is more difficult for them and tedious. Furthermore, it is likely that most users will not know which information is potentially relevant and, therefore, will not know what information to announce. Instead, our approach to context-aware application development is to collect contextual information through automated means, make it easily available to a computer's run-time environment, and let the application designer decide what information is relevant and how to deal with it. This is the better approach, for it removes the need for users to make all information explicit and it puts the decisions about what is relevant into the designer's hands. The application designer should have spent considerably more time analyzing the situations under which the application will be executed and can more appropriately determine what information is relevant and how to react to it.

How is this discussion of context relevant to handheld and ubiquitous computing? In these settings, the user has increased freedom of mobility. The increase in mobility creates situations where the user's context, such as the location of a user and the people and objects around her, is more dynamic. Both handheld and ubiquitous computing have given users the expectation that they can access information services whenever and wherever they are. With a wide range of possible user situations, we need to have a way for the services to adapt appropriately, in order to best support the human–computer interaction.

Realizing the need for context is only the first step toward using it effectively. Most researchers have a general idea about what context is and use that general idea to guide their use of it. However, a vague notion of context is not sufficient; in order to effectively use context, we must attain a better understanding of what context is.

In this paper, we will review previous attempts to define and provide a characterization of context and context-aware computing, and present our own definition and characterization. We then discuss how this increased understanding informs the development of a shared infrastructure for context-sensing and context-aware application development.

2 What is Context?

To develop a specific definition that can be used prescriptively in the context-aware computing field, we will look at how researchers have attempted to define context in their own work. While most people tacitly understand what context is, they find it hard to elucidate. Previous definitions of context are done by enumeration of examples or by choosing synonyms for context.

2.1 Previous Definitions of Context

In the work that first introduces the term 'context-aware,' Schilit and Theimer [26] refer to context as location, identities of nearby people and objects, and changes to those objects. In a similar definition, Brown *et al.* [3] define context as location, identities of the people around the user, the time of day, season, temperature, etc. Ryan *et al.* [22] define context as the user's location, environment, identity and time. Dey [8] enumerates context as the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment. These definitions that define context by example are difficult to apply. When we want to determine whether a type of information not listed in the definition is context or not, it is not clear how we can use the definition to solve the dilemma.

Other definitions have simply provided synonyms for context; for example, referring to context as the environment or situation. Some consider context to be the user's environment, while others consider it to be the application's environment. Brown [4] defined context to be the elements of the user's environment that the user's computer knows about. Franklin & Flaschbart [14] see it as the situation of the user. Ward *et al.* [27] view context as the state of the application's surroundings and Rodden *et al.* [20] define it to be the application's setting. Hull *et al.* [15] included the entire environment by defining context to be aspects of the current situation. As with the definitions by example, definitions that simply use synonyms for context are extremely difficult to apply in practice.

The definitions by Schilit *et al.* [25], Dey *et al.* [9] and Pascoe [17] are closest in spirit to the operational definition we desire. Schilit *et al.* claim that the important aspects of context are: where you are, who you are with, and what resources are nearby. They define context to be the constantly changing execution environment. They include the following pieces of the environment:

- *Computing environment* available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing
- *User environment* location, collection of nearby people, and social situation
- *Physical environment* lighting and noise level

Dey *et al.* define context to be the user's physical, social, emotional or informational state. Finally, Pascoe defines context to be the subset of physical and conceptual states of interest to a particular entity. These definitions are too specific. Context is all about the whole situation relevant to an application and its set of users. We cannot enumerate which aspects of all situations are important, as this will change from situation to situation. In some cases, the physical environment may be important, while in others it may be completely immaterial. For this reason, we could not use the definitions provided by Schilit *et al.*, Dey *et al.*, or Pascoe.

2.2 Our Definition of Context

> *Context is any information that can be used to characterize the situation of an entity. 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 applications themselves.

This definition makes it easier for an application developer to enumerate the context for a given application scenario. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context. Take the task of using a spreadsheet to add a list of weights as an example. The entities in this example are the user and the application. We will look at two pieces of information – presence of other people and location – and use the definition to determine whether either one is context. The presence of other people in the room does not affect the user or the application for the purpose of this task. Therefore, it is not context. The user's location, however, can be used to characterize the user's situation. If the user is located in the United States, the sum of the weights will be presented in terms of pounds and ounces. If the user is located in Canada, the sum of the weights will be presented in terms of kilograms. Therefore, the user's location is context because it can be used to characterize the user's situation.

A general assumption is that context consists only of implicit information, but this is a troublesome distinction. Our definition allows context to be either explicitly or implicitly indicated by the user. For example, whether a user's identity is detected implicitly using vision or explicitly via a login dialogue, the user's identity is still context. Researchers have focussed on implicit information because there is much unexplored promise in leveraging off implicit information pertaining to the human–computer interaction.

2.3 Categories of Context

A categorization of context types will help application designers uncover the most likely pieces of context that will be useful in their applications. Previous definitions of context seed our development of context types. Ryan *et al.* [22] suggest context types of location, environment, identity and time. Schilit *et al.* [25] list the important aspects of context as where you are, who you are with and what resources are nearby.

Context-aware applications look at the *who*'s, *where*'s, *when*'s and *what*'s (that is, what the user is doing) of entities and use this information to determine *why* the situation is occurring. An application doesn't actually determine why a situation is occurring, but the designer of the application does. The designer uses incoming context to determine why a situation is occurring and uses this to encode some action in the application. For example, in a context-aware tour guide, a user carrying a handheld computer approaches some interesting site resulting in information relevant to the site being displayed on the computer. In this situation, the designer has encoded the understanding that when a user approaches a particular site (the 'incoming context'), it means that the user is interested in the site (the 'why') and the application should display some relevant information (the 'action').

There are certain types of context that are, in practice, more important than others. These are ***location***, ***identity***, ***activity*** and ***time***. The only difference between this list and the definition of context provided by Ryan *et al.* is the use of 'activity' rather

than ‘environment’. Environment is a synonym for context and does not add to our investigation of context. Activity, on the other hand, answers a fundamental question of what is occurring in the situation. The categories provided by Schilit *et al.* (where you are, who you are with, and what objects are around you) only include location and identity information. To characterize a situation, we also need activity and time information.

Location, identity, time, and activity are the *primary* context types for characterizing the situation of a particular entity. These context types not only answer the questions of who, what, when, and where, but also act as indices into other sources of contextual information. For example, given a person’s identity, we can acquire many pieces of related information such as phone numbers, addresses, email addresses, birthdate, list of friends, relationships to other people in the environment, etc. With an entity’s location, we can determine what other objects or people are near the entity and what activity is occurring near the entity. From these examples, it should be evident that the primary pieces of context for one entity can be used as indices to find *secondary* context (e.g., the email address) for that same entity as well as primary context for other related entities (e.g., other people in the same location).

In this initial categorization, we have a simple two-tiered system. The four primary pieces of context already identified are on the first level. All the other types of context are on the second level. The secondary pieces of context share a common characteristic: they can be indexed by primary context because they are attributes of the entity with primary context. For example, a user’s phone number is a piece of secondary context and it can be obtained by using the user’s identity as an index into an information space like a phone directory. There are some situations in which multiple pieces of primary context are required to index into an information space. For example, the forecasted weather is context in an outdoor tour guide that uses the information to schedule a tour for users. To obtain the forecasted weather, both the location for the forecast and the date of the desired forecast are required.

This characterization helps designers choose context to use in their applications, structure the context they use, and search out other relevant context. The four primary pieces of context indicate the types of information necessary for characterizing a situation and their use as indices provide a way for the context to be used and organized. Now that we have given a definition of context, we can begin to think about how to use context. In the next section, we define context-aware and provide a characterization of context-aware application features.

3 Defining Context-Aware Computing

Context-aware computing was first discussed by Schilit and Theimer [26] in 1994 to be software that “adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time.” However, it is commonly agreed that the first research investigation of context-aware computing was the Olivetti Active Badge [28] work in 1992. Since then, there have been numer-

ous attempts to define context-aware computing, and these all inform our own definition.

3.1 Previous Definitions of Context-Aware

The first definition of context-aware applications given by Schilit and Theimer [26] restricted the definition from applications that are simply informed about context to applications that *adapt* themselves to context.

Context-aware has become somewhat synonymous with other terms: adaptive [2], reactive [6], responsive [12], situated [15], context-sensitive [19] and environment-directed [13]. Previous definitions of context-aware computing fall into two categories: using context and adapting to context.

We will first discuss the more general case of using context. Hull *et al.* [15] and Pascoe *et al*. [17,18,22] define context-aware computing to be the ability of computing devices to detect and sense, interpret and respond to aspects of a user's local environment and the computing devices themselves. Dey [8] limits context-awareness to the human–computer interface, as opposed to the underlying application. Dey *et al*. [9] begin to introduce the notion of adaptation by defining context-awareness to be work leading to the automation of a software system based on knowledge of the user's context. Salber *et al* [23] define context-aware to be the ability to provide maximum flexibility of a computational service based on real-time sensing of context.

The following definitions are in the more specific "adapting to context" category. Many researchers [25,27,16,3,7,10,1] define context-aware applications to be applications that dynamically change or adapt their behavior based on the context of the application and the user. More specifically, Ryan [21] defines context-aware applications to be applications that monitor input from environmental sensors and allow users to select from a range of physical and logical contexts according to their current interests or activities. This definition is more restrictive than the previous one by identifying the method in which applications acts upon context. Brown [5] defines context-aware applications as applications that automatically provide information and/or take actions according to the user's present context as detected by sensors. He also takes a narrow view of context-aware computing by stating that these actions can take the form of presenting information to the user, executing a program according to context, or configuring a graphical layout according to context. Finally, Fickas *et al* [13] define environment-directed (practical synonym for context-aware) applications to be applications that monitor changes in the environment and adapt their operation according to predefined or user-defined guidelines.

3.2 Our Definition of Context-Aware

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*

We have chosen a more general definition of context-aware computing. The definitions in the more specific "adapting to context" category require that an application's behavior be modified for it to be considered context-aware. When we try to apply these definitions to established context-aware applications, we find that they do not fit. For example, an application that simply displays the context of the user's environment to the user is not modifying its behavior, but it is context-aware. If we use the less general definitions, these applications would not be classified as context-aware. We, therefore, chose a more general definition that does not exclude existing context-aware applications.

We will compare our definition to the general definitions provided above. Our definition is more general than the one provided by Hull *et al.* [15] and Pascoe *et al.* [17,18,22]. They require their context-aware systems to detect, interpret and respond to context. We only require the response to context, allowing the detection and interpretation to be performed by other computing entities. It differs from the other general definitions given above by focussing on the user, not limiting awareness to just the application interface [8], not requiring applications to perform services automatically [9], and not requiring real-time acquisition of context [23].

Categorization of Features for Context-Aware Applications In a further attempt to help define the field of context-aware applications, we will present a categorization for features of context-aware applications. There have been two attempts to develop such a taxonomy. The first was by Schilit *et al.* [25] and had 2 orthogonal dimensions: whether the task is to get information or to execute a command and whether the task is executed manually or automatically. Applications that retrieve information for the user manually based on available context are classified as *proximate selection* applications. It is an interaction technique where a list of objects (printers) or places (offices) is presented, where items relevant to the user's context are emphasized or made easier to choose._Applications that retrieve information for the user automatically based on available context are classified as *automatic contextual reconfiguration*. It is a system-level technique that creates an automatic binding to an available resource based on current context. Applications that execute commands for the user manually based on available context are classified as *contextual command* applications. They are executable services made available due to the user's context or whose execution is modified based on the user's context. Finally, applications that execute commands for the user automatically based on available context use *context-triggered actions*. They are services that are executed automatically when the right combination of context exists, and are based on simple if-then rules.

More recently, Pascoe [17] proposed a taxonomy of context-aware features. There is considerable overlap between the two taxonomies but some crucial differences as well. Pascoe [17] developed a taxonomy aimed at identifying the core features of context-awareness, as opposed to the previous taxonomy, which identified classes of context-aware applications. In reality, the following features of context-awareness map well to the classes of applications in the Schilit taxonomy. The first feature is *contextual sensing* and is the ability to detect contextual information and present it to the user, augmenting the user's sensory system. This is similar to *proximate selection*,

except in this case, the user does not necessarily need to select one of the context items for more information (i.e. the context may be the information required). The next feature is *contextual adaptation* and is the ability to execute or modify a service automatically based on the current context. This maps directly to Schilit's *context-triggered actions*. The third feature, *contextual resource discovery*, allows context-aware applications to locate and exploit resources and services that are relevant to the user's context. This maps directly to *automatic contextual reconfiguration*. The final feature, *contextual augmentation*, is the ability to associate digital data with the user's context. A user can view the data when he is in that associated context. For example, a user can create a virtual note providing details about a broken television and attach the note to the television. When another user is close to the television or attempts to use it, he will see the virtual note left previously.

Pascoe and Schilit both list the ability to exploit resources relevant to the user's context, the ability to execute a command automatically based on the user's context and the ability to display relevant information to the user. Pascoe goes further in terms of displaying relevant information to the user by including the display of context, and not just information that requires further selection (e.g. showing the user's location vs. showing a list of printers and allowing the user to choose one). Pascoe's taxonomy has a category not found in Schilit's taxonomy: *contextual augmentation*, or the ability to associate digital data with the user's context. Finally, Pascoe's taxonomy does not support the presentation of commands relevant to a user's context. This presentation is called *contextual commands* in Schilit's taxonomy.

Our proposed categorization combines the ideas from these two taxonomies and takes into account the three major differences. Similar to Pascoe's taxonomy, it is a list of the context-aware features that context-aware applications may support. There are three categories:

1) *presentation* of information and services to a user;
2) automatic *execution* of a service; and
3) *tagging* of context to information for later retrieval

Presentation is a combination of Schilit's proximate selection and contextual commands. To this, we have added Pascoe's notion of presenting context (as a form of information) to the user. Automatic execution is the same as Schilit's context-triggered actions and Pascoe's contextual adaptation. Tagging is the same as Pascoe's contextual augmentation.

We introduce two important distinguishing characteristics: the decision not to differentiate between information and services, and the removal of the exploitation of local resources as a feature. We do not to use Schilit's dimension of information vs. services to distinguish between our categories. In most cases, it is too difficult to distinguish between a presentation of information and a presentation of services. For example, Schilit writes that a list of printers ordered by proximity to the user is an example of providing information to the user. But, whether that list is a list of information or a list of services depends on how the user actually uses that information. For example, if the user just looks at the list of printers to become familiar with the names of the printers nearby, she is using the list as information. However, if the user chooses a printer from that list to print to, she is using the list as a set of services.

Rather than try to assume the user's state of mind, we chose to treat information and services in a similar fashion.

We chose not to use the exploitation of local resources, or resource discovery, as a context-aware feature. This feature is called automatic contextual reconfiguration in Schilit's taxonomy and contextual resource discovery in Pascoe's taxonomy. We do not see this as a separate feature category, but rather as part of our first two categories. Resource discovery is the ability to locate new services according to the user's context. This ability is really no different than choosing services based on context. We can illustrate our point by reusing the list of printers example. When a user enters an office, their location changes and the list of nearby printers changes. The list changes by having printers added, removed, or being reordered (by proximity, for example). Is this an instance of resource exploitation or simply a presentation of information and services? Rather than giving resource discovery its own category, we split it into two of our existing categories: presenting information and services to a user and automatically executing a service. When an application presents information to a user, it falls into the first category, and when it automatically executes a service for the user, it falls into the second category.

Cataloguing previous context-aware work We applied our categories of context and context-aware features to the applications discussed in the related research we have presented. The results are in Table 1 below. Under the context type heading, we present Activity, Identity, Location, and Time. Under the context-aware heading, we present our 3 context-aware features, Presentation, automatic Execution, and Tagging.

Table 1. Application of context and context-aware categories

System Name	System Description	Context Type				Context-Aware		
		A	I	L	T	P	E	T
Classroom 2000 [1]	Capture of a classroom lecture			X	X			X
Cyberguide [1]	Tour guide		X	X		X		
Teleport [2]	Teleporting	X	X	X			X	
Stick-e Documents [3,4,5]	Tour guide		X	X	X	X		X
	Paging and reminders	X	X			X		X
Reactive Room [6]	Intelligent control of audiovisuals	X	X	X			X	
GUIDE [7]	Tour guide			X		X		
CyberDesk [8,9,10]	Automatic integration of user services	X				X	X	
Conference Assistant [11]	Conference capture and tour guide	X	X	X	X	X		X
Responsive Office [12]	Office environment control			X	X		X	
NETMAN [13,16]	Network maintenance			X		X		
Fieldwork [17,18,22]	Fieldwork data collection			X	X	X		X
Augment-able Reality [19]	Virtual post-it notes			X		X		X
Context Toolkit [24]	In/Out Board		X	X	X	X		
	Capture of serendipitous meetings		X	X	X		X	X
Active Badge [28]	Call forwarding		X	X		X	X	

From this table, we see that while many applications make use of both location and identity context, few make use of activity context. We also note that many applications use the presentation feature, but fewer use the execution and tagging features. This table indicates that there are areas which need addressing in the space of context-aware applications.

4 Impact on Context-Aware Application Development

In the previous sections, we presented definitions and categories for both context and context-aware. We have applied these definitions and categories to our own research and they have helped to inform the design of an architecture to support context-aware computing and the design of context-aware applications [11].

Our architecture[1] is built on the concept of enabling applications to obtain the context they require without them having to worry about how the context was sensed. In previous work, we presented the context widget [24], an abstraction that implements this concept. A context widget is responsible for acquiring a certain type of context information and it makes that information available to applications in a generic manner, regardless of how it is actually sensed.

Our definition of context provided another important abstraction. We defined it as information used to characterized the situation of an entity. If we think of a context widget as being responsible for a single piece of information, we need an abstraction to represent an entity. This abstraction, which we call a context server in our architecture, is responsible for the entire context about a single entity. When designers think about context and interactions, it is natural for them to think in terms of entities, and that makes a context server the correct abstraction to use for building applications. Context servers gather the context about an entity (e.g., a person) from the available context widgets, behaving as a proxy to the context for applications.

Context types inform us both on the variety of context widgets needed and on the features needed to be supported by context servers. At the very minimum, our architecture needs to contain context widgets that collect information on our four primary context types: activity, identity, location, and time. Our context servers also need to support these primary types. Primary context types could be used to both index into more information about a particular entity (secondary context) and index into other entities. To allow applications to leverage this first feature, context servers need to support the retrieving of secondary context information indexed by primary context types. For the second feature, context servers must provide facilities to each other to allow comparison on the primary context types. This will allow an entity to create dynamic relationships with and locate other entities that share all or part of its context, for example, to identify all the people meeting in the same room.

While our definition of context-aware has not directly impacted our architecture, it has provided us with a way to conclude whether an application is context-aware or not. This has been useful in determining what types of applications our architecture

[1] For more information on the architecture, see http://www.cc.gatech.edu/fce/contexttoolkit

needs to support, and therefore what features it must have. Our categorization of context-aware features provides us with two main benefits. The first is that it further specifies the types of applications the architecture must provide support for. The second benefit is that it shows us the types of features that we should be thinking about when building our own context-aware applications.

5 Conclusions

In handheld and ubiquitous computing, a user's context is very dynamic. When using applications in these settings, a user has much to gain by the effective use of implicitly sensed context. It allows an application's behavior to be customized to the user's current situation. To promote a more effective use of context, we have provided definitions and categorizations of context and context-aware computing.

We have defined context to be any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or object. In context-aware computing, these entities are anything relevant to the interaction between the user and application, including the user and the application. We provided an initial set of primary context types —location, identity, time and activity. Context-aware applications use context to provide task-relevant information and/or services to a user. We defined categories of context-aware applications —the presentation of information and services, automatic execution of services, and tagging information with context.

These definitions and categorizations will assist researchers understand the boundaries of context-aware computing, and help application designers select context to use, structure context in applications, and decide what context-aware features to implement. In an effort to substantiate this claim, we have begun work on a framework [24] to support the building of context-aware applications. The definitions and categorizations presented in this paper have both influenced the design of the framework [11] and the design of context-aware applications we have built. Through the use of this framework and the development of more applications, we hope to further increase our understanding of context and context-awareness.

References

1. Abowd, G.D., Dey, A.K., Orr, R., Brotherton, J. Context-Awareness in Wearable and Ubiquitous Computing. 1st International Symposium on Wearable Computers (1997) 179-180
2. Brown, M. Supporting User Mobility. International Federation for Information Processing (1996)
3. Brown, P.J., Bovey, J.D. Chen, X. Context-Aware Applications: From the Laboratory to the Marketplace. IEEE Personal Communications, 4(5) (1997) 58-64
4. Brown, P.J. The Stick-e Document: a Framework for Creating Context-Aware Applications. Electronic Publishing '96 (1996) 259-272
5. Brown, P.J. Triggering Information by Context. Personal Technologies, 2(1) (1998) 1-9
6. Cooperstock, J., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W. Evolution of a Reactive Environment CHI '95 (1995) 170-177

7. Davies, N., Mitchell, K., Cheverst, K. Blair, G. Developing a Context Sensitive Tourist Guide. 1st Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1 (1998)
8. Dey, A.K. Context-Aware Computing: The CyberDesk Project. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 51-54
9. Dey, A.K., Abowd, G.D., Wood, A. CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. Knowledge-Based Systems, 11 (1999) 3-13
10. Dey, A.K., Abowd, G.D. CyberDesk: The Use of Perception in Context-Aware Computing. 1st Workshop on Perceptual User Interfaces (1997) 26-27
11. Dey, A.K., Salber, D., Futakawa, M., Abowd, G.D. An Architecture to Support Context-Aware Computing. Submitted to UIST '99
12. Elrod, S., Hall, G., Costanza, R., Dixon, M., des Rivieres, J. Responsive Office Environments. CACM 36(7) (1993) 84-85
13. Fickas, S., Korteum, G., Segall, Z. Software Organization for Dynamic and Adaptable Wearable Systems. 1st International Symposium on Wearable Computers (1997) 56-63
14. Franklin, D., Flaschbart, J. All Gadget and No Representation Makes Jack a Dull Environment. AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 155-160
15. Hull, R., Neaves, P., Bedford-Roberts, J. Towards Situated Computing. 1st International Symposium on Wearable Computers (1997) 146-153
16. Korteum, G., Segall, Z., Bauer, M. Context-Aware, Adaptive Wearable Computers as Remote Interfaces to "Intelligent" Environments. 2nd International Symposium on Wearable Computers (1998) 58-65
17. Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. 2nd International Symposium on Wearable Computers (1998) 92-99
18. Pascoe, J., Ryan, N.S., Morse, D.R. Human-Computer-Giraffe Interaction – HCI in the Field. Workshop on Human Computer Interaction with Mobile Devices. (1998)
19. Rekimoto, J., Ayatsuka, Y., Hayashi, K. Augment-able Reality: Situated Communication through Physical and Digital Spaces. 2nd International Symposium on Wearable Computers (1998) 68-75
20. Rodden, T., Cheverst, K., Davies, K. Dix, A.. Exploiting Context in HCI Design for Mobile Systems. Workshop on Human Computer Interaction with Mobile Devices (1998)
21. Ryan, N. Mobile Computing in a Fieldwork Environment: Metadata Elements. Project working document, version 0.2 (1997)
22. Ryan, N., Pascoe, J., Morse, D. Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology (1997)
23. Salber, D., Dey, A.K., Abowd, G.D. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. Georgia Tech GVU Technical Report GIT-GVU-98-01 (1998)
24. Salber, D., Dey, A.K., Abowd, G.D. The Context Toolkit: Aiding the Development of Context-Enabled Applications. To appear in CHI'99 (1999)
25. Schilit, B., Adams, N. Want, R. Context-Aware Computing Applications. 1st International Workshop on Mobile Computing Systems and Applications. (1994) 85-90
26. Schilit, B., Theimer, M. Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) 22-32
27. Ward, A., Jones, A., Hopper, A. A New Location Technique for the Active Office. IEEE Personal Communications 4(5) (1997) 42-47
28. Want, R., Hopper, A., Falcao, V., Gibbons, J. The Active Badge Location System. ACM Transactions on Information Systems 10(1) (1992) 91-102