Implicit Human Computer Interaction Through Context

Albrecht Schmidt

Telecooperation Office (TecO), University of Karlsruhe, Karlsruhe, Germany

Abstract: In this paper the term "implicit human-computer interaction" is defined. It is discussed how the availability of processing power and advanced sensing technology can enable a shift in HCI from explicit interaction, such as direct manipulation GUIs, towards a more implicit interaction based on situational context. In the paper, an algorithm is given based on a number of questions to identify applications that can facilitate implicit interaction. An XML-based language to describe implicit HCI is proposed. The language uses contextual variables that can be grouped using different types of semantics as well as actions that are called by triggers. The term of perception is discussed and four basic approaches are identified that are useful when building context-aware applications. Two examples, a wearable context awareness component and a sensor-board, show how sensor-based perception can be implemented. It is also discussed how situational context can be exploited to improve input and output of mobile devices.

Keywords: Context awareness; Context sensing; Implicit human-computer interaction; Perception; Ubiquitous computing

1. Introduction

The way people interact with devices is vital for their success. Looking at HCI, it is apparent that interaction techniques are limited by the technology available. Furthermore, the anticipated user groups influence the interaction metaphors to a large extent. Considering the shift from punch cards to interactive text terminals and also the shift from command line interfaces to graphical user interfaces (GUI), this was observable.

Bearing in mind current and upcoming technologies, such as increased processing power (even on mobile devices), availability of sensors (ranging from simple temperature sensors to cameras), and the resulting perceptional capabilities as well as the fact that the main user group of current computing devices (e.g. mobile phones, PDAs, etc.) is non expert, we may observe yet another shift in HCI. Devices that have perceptional capabilities (even if they are very limited) will start the shift from explicit HCI towards a more implicit interaction with machines.

A vision of future devices: We will be able to create (mobile) devices that can see, hear and feel. Based on their perception, these devices will be able to act and react according to the situational context in which they are used.

In this paper it will be shown that this vision is not as far ahead as it seems. In our research we start with the perception of simple concepts and with their exploitation. Providing a number of examples and demonstrators it is discussed how basic perception could enable a shift from explicit towards implicit HCI.

2. Implicit Interaction

When observing communication between humans, we can see that a lot of information is only exchanged implicitly. The way people interact with each other, and also the situation in which they interact, carries information that is often implicitly exploited in the exchange of messages. While in a conversation, both the behaviour of participants and what is happening in the surrounding environment supply valuable information that is often vital for the understanding of messages. In many cases the robustness of human-to-human communication is based on the implicitly introduced contextual information, such as gestures, body language, and voice. Another example is redundancy between body language (e.g. nodding) and spoken language (e.g. the word "yes"). This implicitly introduced knowledge is also used to disambiguate information, e.g. in a discussion with a student pointing at a computer the term "sun" has a different meaning than the same term when on the beach together with friends; a more in depth discussion is given in [1].

Personal Technologies (2000) 4:191-199

2.1. Implicit vs. explicit human-computer interaction

Current computer technology interaction is explicit – the user tells the computer in a certain level of abstraction (e.g. by command-line, direct manipulation using a GUI, gesture, or speech input) what she expects the computer to do. This is considered as explicit interaction.

Definition of implicit human-computer interaction: an action performed by the user that is not primarily aimed to interact with a computerised system but which such a system understands as input.

The action of a user is always performed in a certain environment. Implicit interaction is based on the assumption that the computer has a certain understanding of our behaviour in the given situation. This knowledge is then considered as an additional input to the computer while doing a task.

A simple example is the garbage bin [2] that scans in the bar code of products and reproduces the information for a suggested shopping list. The action performed by the user (e.g. throw an empty can into a bin) is the same as with any other garbage bin. The recognition of the system (by scanning the bar code) and the built-in interpretation of the system (all things that go into the bin may be on the next shopping list) make use of the action performed by the user. The user herself does not explicitly interact with the computer, thus the process describes an implicit interaction. As we see from the example implicit interaction is based on two main concepts:

- perception
- interpretation

For most applications, implicit interaction will be used additionally to explicit interaction.

Other systems are implemented that also facilitate the idea of implicit interaction on a rudimentary level, e.g. automatic light control (switches on the light when it is dark and someone is walking by), and active badge systems (automatically open a door when someone with appropriate permission wants to enter the building). In current computer systems we can observe that agent technology is used to build systems that have a certain ability to act pro-actively. These approaches are mainly based on user profiles and usage information [3]. In these cases perception is limited to information gathered in the virtual space.

If we look at concepts that are needed to facilitate implicit interaction three basic building blocks can be identified:

1. the ability to have perception of the use, the environment, and the circumstances;
2. mechanisms to understand what the sensors see, hear and feel; and
3. applications that can make use of this information.

On a conceptual level (1) and (2) can be described as situational context, and (3) are applications that are context enabled. In the next section context is discussed in more detail.

2.2. What is context?

The notion of context is used in many different ways. In our work we propose to regard situational context, such as location, surrounding environment or state of the device, as implicit input to the system. We use the term "situational context" to describe implicit interaction fragments. This extends the concept of context beyond the informational context into real-world environments.

In general use the word "context" has a multitude of meanings. Even within the field of computer science different disciplines, such as artificial intelligence, natural language processing, image recognition, and more recently mobile computing, have their very own understanding of what context is. In our work we found that very general descriptions of context as given by a dictionary and also synonyms found in a thesaurus come very close to our understanding. To illustrate this we want to provide the following definitions:

Context n 1: discourse that surrounds a language unit and helps to determine its interpretation [syn: linguistic context, context of use] 2: the set of facts or circumstances that surround a situation or event; "the historical context" (Source: WordNet ® 1.6)

Context: That which surrounds, and gives meaning to, something else.
(Source: The Free On-line Dictionary of Computing)

Synonyms: Circumstance, situation, phase, position, posture, attitude, place, point; terms; régime; footing, standing, status, occasion, surroundings, environment, location, dependence.
(Source: www.thesaurus.com)

To build applications which have knowledge about their situational context it is important to gain an understanding of what context is. Current research in context-awareness in mobile computing shows a strong focus on location [4-7]. Location is a concept that is well understood. Also the benefit

of location-awareness is clearly given; at certain locations particular services are more important than others. An architectural approach based on a smart environment is described by Schilit et al. [6]. Other scenarios are using RF and GPS to determine the users' location, e.g. [7,8]. But, as pointed out in [9], context is more than location. We use the term "context" as applied to mobile computing in a more general way, as also suggested by [10], to describe the environment, situation, state, surroundings, task, and so on. A wider view of context is also given by [11]. They suggest that the way a device is used (mobile phone in the users hand, on the table, in pocket, etc.) to be considered as context.

2.3. Applications in context

Analysing the way people use ultra-mobile devices (e.g. personal digital assistants, smart mobile phones, handheld and wearable computers), it becomes apparent that the periods of interaction are much shorter than in traditional mobile settings. Notebooks – considered as mobile computers – are used mainly in stationary setting, e.g. one takes a notebook to a meeting and takes notes and a salesman takes a mobile computer to a customer for a presentation. In general in these scenarios, the application is used in a stationary setting between several minutes and hours. Whereas considering the usage of ultra-mobile devices, interaction periods are often much shorter, e.g. looking up an address takes only a few seconds and making a note on a PDA is often in the range of several seconds up to some minutes. Also, the fact that the applications are mainly used while doing something else or to carry out a certain task (like tools in the real world) calls for a reduction of the explicit human-machine interaction and creates the need to shift towards implicit HCI.

Knowledge about the situational context is of primary interest to the application, because we consider that the application will adapt to the context.

It can be observed that an application (mobile or stationary alike) is:

(a) running on a specific device (e.g. input system, screen size, network access, portability, etc.)
(b) at a certain time (absolute time, e.g. 9:34 pm; class of time, e.g. in the morning)
(c) used by one or more users (concurrently or sequentially)
(d) in a certain physical environment (absolute location, type of location, conditions such as light, audio, and temperature, infrastructure, etc.)
(e) in a social setting (people colocated and social role)
(f) to solve a particular task (single task, group of tasks, or a general goal)

We consider the items (a) to (f) as the basic building blocks of context. For mobile applications especially, (d) and (e) are of major interest. In mobile settings the physical environment can change while an application is executed, e.g. making a phone call while walking from the office desk to the car park. The telephone application is running, while the noise level changes between office and outside.

2.4. Identifying implicit human-computer interaction

Analysis must be carried out to identify an application that can be improved by implicit HCI input and output of the application and the real-world environment in which it is executed. Then ways to capture the situational context must be assessed. Furthermore, mechanisms for the interpretation of the situational context have to be found. Finally the reaction of the application has to be defined. The following questions help to identify these points:

- What happens around an application while the application is in use? Are there any changes at all?
- Do the surroundings (behaviour, environment, circumstances) carry any valuable information for the application? Does it matter for the application?
- Are there any means to capture and extract the information in a way that is acceptable for the application or device (processing cost, sensor cost, weight, etc.)?
- How to understand the information? What interpretation and reasoning is possible and useful? What is an appropriate way for the application to react?

Putting all of these together we can set up the algorithm in Fig. 1. The algorithm works as follows:

On 1: $\mathbf{C}$ is the set of surrounding conditions which carries information that is useful for the application. Each element C_i stands for one condition, e.g. location, temperature, current user, device orientation, etc. The set is created by asking what conditions are changing in the environment.

```
1. create the set C
2. set D = {}
3. for each C_i ∈ C
      define A_i. // accuracy
      define U_i. // update rate
      identify S_i // a sensor device
                   // that is appropriate
      if cost(S_i, A_i, U_i) is acceptable then
            D = D ∪{ (C_i, S_i, A_i, U_i) }
      fi
   next
4. if D ≠ {} then
      for each vector D_i in D
            define a set of application reaction R_i = { (I_ij, R_ij) }
            // I_ij is input range, application reaction pairs  I_ij
            // R_ij is application reaction
   else
      // implicit interaction is not used
      //(either no condition that are useful or too costly)
```

Fig. 1. Identifying Implicit HCI.

On 2: D is initialised – at the beginning no sensing devices are identified.

On 3: For each C_i the accuracy A_i and the update rate U_i that are needed to make the measurements useful are defined. Then a sensing device that matches these requirements is identified. If the cost for the identified sensing device D_i is acceptable, then the vector describing the condition, the sensing device, the required accuracy and update rate is added to the set D. For conditions that cannot be sensed the cost is infinite.

On 4: If any conditions that are feasible to measure exist then for each of these conditions one or more range values that are meaningful (temperature between 15°C and 25°C, location is inside my office, user is moving, etc.) are identified and for these ranges the reaction of the application (switch to notepad, etc.) is defined.

2.5. Modelling implicit human-computer interaction

To specify applications that facilitate implicit HCI, it is inevitable that a specification language is needed to describe situational context linked to events/changes that occur in the application. In our recent work we found it helpful to use a notation that is easy for humans to read, as well as easy to process using a computer. We decided to use a markup language that is specified in XML for this purpose. Extending the SGML based description model introduced by Brown in [10,12], we added two more concepts – grouping context with matching attributes and trigger attributes to make the description more expressive and suitable for our projects (see Fig. 2 for the XML data type definition DTD).

In the `<context>` section contextual variables are used to describe the conditions. These variables are made of two parts. The first is used to specify the context sensing module (Fig. 5), the sensor module (`sensor_module`) and the palm pilot (`pilot`), and in the second part the variables provided by this module.

In the `<action>` section, function calls are used to specify the action to be carried out in case the trigger evaluates to true. These calls are also hierarchically structured; specifying the device, the application, and the function to be performed.

Depending on the platform (e.g. context sensing module in a microcontroller) we use a different implementation language.

```
<!ELEMENT context_interaction
              (context , action )>
<!ELEMENT context  (group+ )>
<!ELEMENT group  (#PCDATA )>
<!ATTLIST group  match
   (one | all | none ) #REQUIRED >
<!ELEMENT action  (#PCDATA )>
<!ATTLIST action
    time  CDATA  '0'
    trigger  (enter | leave | in )
                      #REQUIRED >
```

Fig. 2. Data Type Definition.

```
<context_interaction>
  <context>
     <group match='one'>
        sensor_module.touch
        pilot.on
     </group>
     <group match='none'>
        sensor_module.alone
        pilot.pen_down
     </group>
  </context>
  <action trigger='enter' time='3'>
     pilot.notepad.confidential
  </action>
</context_interaction>
```

Fig. 3. Context description.

If contexts are composed of a number of components we found it very helpful to have a mechanism to bundle certain contextual variables in groups and select a matching semantic for each group description. For matching in a group we provided the following semantics: one (match one or more of the variables in the following group); all (match all variables in the following group); none (match none of the variables in the following group). All groups within the context description must evaluate to true to cause the trigger.

We discriminate three different triggers: "enter a context", "leave a context", and "while in a context". The "enter" and "leave" triggers take a time value that specifies the time after[1] which the action is triggered if the context stays stable over this time. For the "while in a context" trigger the time indicates the interval in which the trigger is fired again.

In Fig. 3 an example of a description of a context and an action is shown. The context description consists of two groups of contextual variables. In the first group, the match semantics is that at least one of the variables must be true – in this case either the device is touched or the state of the device is on. In the second group the match semantics is "none", which means that the contextual variable alone must not be true and that the user must not have touched the screen with a pen.

If the context evaluates to true, an action is triggered. Here the semantics is that if the context is entered and is stable for at least three seconds then the action is performed.

The complete description means that if the device is on or in the user's hand and if the user is not alone and is not touching the screen with the pen then after three seconds the display should be hidden by an image as depicted in Fig. 6 (d) below.

3. Perception

There are several ways to equip devices with perceptional capabilities. The range of complexity to consider is very wide, starting from simple sensors that know the way a device is held [13] to complex audio and video analysis. We identified the following four basic approaches:

- device-databases (e.g. calendars, todo-lists, address books, profile, etc.);
- input to the application running (notepad: taking notes; calendar: looking up a date, etc.);
- active environments (active badges [14], IR-networks, cameras, audio, etc.)
- sensing context using sensors (TEA [15, 16], Sensor Badges [4], GPS, cameras, audio [17], etc.).

The perceptual capabilities can be located in the device itself, in the environment or in another device that shares the context over a network (e.g. body area network).

In the remainder of this section we concentrate on sensor-based perception, also knowing that in most scenarios a combination of all four cases is the way of choice. First we introduce two sensor devices developed in our group and then provide

[1] The parameter indicating the time after which an action is performed is often 0 (immediate context action coupling) or positive. In certain circumstances, when future situations can be predicted (e.g. you drive your car into the car park, the context walking will appear soon), a negative value does make sense, too.

some information on other sensor-based devices that supply contextual information.

3.1. Context awareness component

In this Section a wearable context-aware component that integrates low-cost sensors is described. Simple methods are used to derive context information from sensor data. The derived context is application-independent and can be exploited by other wearable or personal technologies in a body network, for instance wearable computers, mobile phones, digital cameras, and personal digital assistants.

Here we chose to address a number of contexts that relate to how easy it is to interrupt the user. These contexts describe only a certain aspect of real-world situations but they are general in the sense that they can be exploited by a range of applications. Such a context is, for instance, implemented and used in the audio wearable described in [17], mimicking the human ability to recognise situations in which it is rude to interrupt, such as when a person is engaged in a conversation or giving a talk. Context information of this kind is also useful for other applications, for instance, calendars, email notification, pagers and mobile phones can make use of any context which gives an indication of whether or not it is a good time to interrupt a user.

The specific contexts that we chose for our study are based on aural information – user speaking, others speaking, noisy, and quiet – and are also based on movement of the user – walking, running, stationary. Movement context was included as it gives an indication as to whether a user can be interrupted visually.

For recognition of aural and movement contexts, we integrated two microphones and an accelerometer in our design. One of the microphones is placed near the user's throat, the other points away from the user. With this configuration the distinction of speaker and environment is feasible with minimal processing cost. The acceleration sensor is used to discriminate whether a user is standing still, walking or running.

The sensor placement considerations led us to build the context-awareness component into a tie. It may be possible to build them into other accessories worn in similar ways (e.g. jewellry, neckerchief, or necklace). We also liked the way that a tie stresses the component's design as accessory rather than as stand-alone device (Fig. 4).

The hardware of our context-awareness component is build around a TINY-Tiger micro-

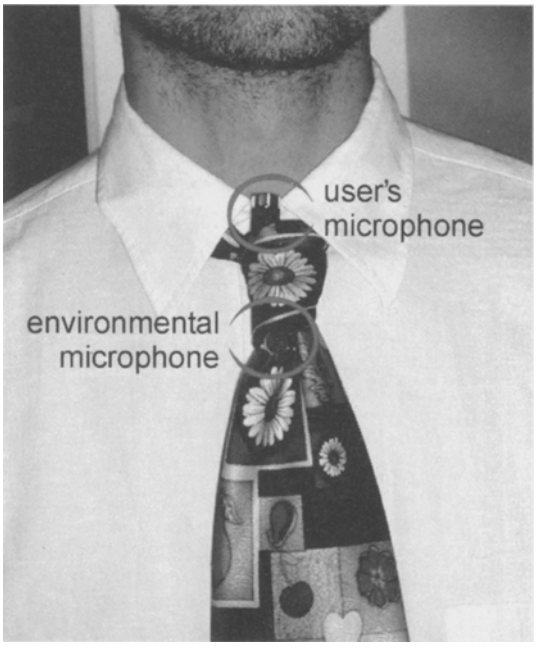

Fig. 4. Contest tie.

controller, which offers four analog inputs and two serial lines. The two signals from the microphones are amplified and connected to the analog inputs. To measure the motion we used a two-axis accelerometer (Analog Devices ADXL202). A more detailed description has been published in [18].

The software is realised in Tiger-BASIC, a multitasking basic dialect for the TINY-Tiger. It reads and analyses sensor data in a time window of about four seconds. The methods to analyse the signals are deliberately simple; they work within the time domain and are based on basic statistical measurements. Based on the features calculated from sensor data the contexts are detected.

The communication is based on a serial line connection using 9600 bit/s, in a simple request–reply manner. The client requests the contextual variables from the context-awareness component that sends back the variables together with the values.

Experimentation with the context-aware tie showed that contexts were recognised in a very reliable way. Both "user speaking" vs. "others speaking" and "stationary" vs. "walking" vs. "running" were discriminated correctly. A key finding is that sensor placement can be used effectively to increase reliability and to reduce required processing.

The device can provide information on the situational context of the user for other personal

Fig. 5. Context Sensing Device and PlamPilot.

technologies in a body area network. Using this device the implicit HCI can be facilitated.

3.2. Sensor board

Using this board we collect data on the situational context by using a combination of low-level sensors. In this project we built a context recognition device equipped with a light sensor, acceleration sensor, a passive infrared sensor, a touch sensor, and a temperature sensor. All sensors except the touch sensor are standard sensors and produce analog voltage level. The touch sensor recognises the human body as a capacitor and supplies a digital value. The heart of the device is a *BASICTiger* microcontroller that reads from all the physical input channels (it offers four analog digital converters and a number of digital IOs) and statistical methods are applied to recognise contexts. The board is depicted in Fig. 5. The PDA requests contextual variable while the application is idle, e.g. catching the NullEvent on the PalmPilot.

3.3. Related work on context sensing

This way of perception is widely used in robotics but with a different objective – giving machines the ability to operate autonomously.

For use with handheld devices, the project TEA [15] developed a sensor board (equipped with eight sensors, light, acceleration, pressure, temperature, etc.) that supplies contextual information; communication is done via serial line. The application described is a mobile phone that recognises its context (in the user's hand, on the table, in a suitcase, outdoors) and it adapts ringing modes according to the user's preferences in that situation [11].

Using a similar approach, a system to facilitate indoor location awareness based on low level sensors is described in [19]. The system reads data from different sensors (acceleration, light, magnetic field, etc.) and provides location information.

In [20] a cup is described that has an acceleration and temperature sensor built in together with a microcontoller and infrared communication. The cup is aware of its state (warm, cold, on a table, drinking, moved). The information from a number of cups communicated to a server is then used to supply information about the group of users. All these projects focus on a totally sensor-based approach to context awareness.

A jacket that knows if it is on the hanger or with the user is presented in [21]. The sensor jacket has woven-in sensors that give information if the user is wearing the jacket, what movements the user is making, etc. As one application, correcting movements in sports (automated tennis coach) is suggested in the paper. In this project the development of robust sensing technology is very central.

4. How Can HCI Benefit from Context?

HCI for mobile devices is concerned with the general trade-off between devices qualities (e.g. small size, light weight, little energy consumption, etc.) and the demand for optimal input–output capabilities. Here implicit HCI can offer interesting alternatives.

4.1. Output in context

Over recent years the output systems for mobile devices have become much better; features such as stereo audio output, high-resolution colour screens for PDAs and even mobile phones, as well as display systems for wearable computers are commercially available. Also unobtrusive notification mechanisms (e.g. vibration) have become widely used in phones and PDAs. Even so, at the lower end devices with very poor display quality are still entering the market. Situational context can help to:

- adapt the output to the current situation (fontsize, volume, brightness, privacy settings, etc.) [11];
- find the most suitable time interruption [17, 18];
- reduce the need for interruptions (e.g. you don't need to remind someone to go to a meeting if they are already there.)

4.2. Input in context

With very small appliances the space for a keyboard is very limited which results in bad usability. Other input systems, such as graffiti and handwriting recognition, have been developed further but still lack speed and accuracy [22]. Advances in voice recognition have been made in recent years, but for non-office settings (e.g. in a car, in a crowded place, sharing rooms with others, and in industry workplaces), the recognition performance is still poor. Also privacy and acceptance issues are a major concern. Implicit HCI does not solve these problems in general but can help to:

- adapt the input system to the current situation (e.g. audio filter, recognition algorithms, etc.);
- limit need for input (e.g. information is already provided by the context and can be captured);
- reduce selection space (e.g. only offer appropriate options in current context).

4.3. Context NotePad on a PalmPilot

To explore ways of implicit communication between users and their environment with mobile devices we built a context-aware NotePad application. The system uses the perceptional capabilities of the sensor board described in the previous Section and provides an application that is very similar in functionality to the built-in notepad application on the PalmPilot. Additionally, the application can adapt to the current situational context and can also react in this way to the implicit interaction. The application changes its behaviour according to the situation. The following context adaptations have been implemented.

- ***On/Off:*** The user has the device in her hand and in this case the application is switched on. Iif the user puts the device down, it is switched off after a certain time. It assumes that if the user takes the device in her hand she wants to work with it.
- ***Fontsize:*** If the device is moved (e.g. while walking or on a bumpy road) the font size is increased to make reading easier. In contrast, when the device is in a stable position (e.g. stationary in your hand or on the table), the font is made smaller to display more text at the same screen (see Fig. 6a and b).
- ***Backlight:*** This adaptation is straightforward but still not built into current PDAs. By monitoring the light condition, the application switches on the backlight if the brightness level in the environment is below a certain threshold. If it becomes brighter, the light is switched off again (Fig. 6c).
- ***Privacy settings:*** If you are not alone and you are not writing (or touching the screen) the content on the display is hidden by an image (Fig 6d). To sense if someone is walking the passive infrared sensor is deployed.

Fig. 6. Adaptation to context a) small font, b) large font, c) backlight, d) privacy.

Currently we are decreasing the size of the context-awareness device to make it feasible to plug it into the pilot to allow proper user studies.

6. Conclusion and Further Work

Based on observations of new sensing technology, available sensors and anticipated users, a new interaction metaphor is proposed. Implicit HCI is defined as an action performed by the user that is not primarily aimed to interact with a computerised system but which such a system understands as input. It is further identified that perception and interpretation of the user, the environment, and the circumstances are key concepts for implicit HCI. Furthermore applications that exploit this information are required.

Perception and interpretation are considered as key issues to facilitate situational contexts. Therefore we motivate a broad view of context, and also suggest that the context is described from the perspective of the application. To identify applications that can make use of situational context and thus can facilitate implicit HCI, a number of questions are raised and an algorithm is suggested. It is based on the central questions: what happens around the application; how can this be sensed or captured; how to interpret this information; and how can applications make use of it.

From current projects we learned that there is a need for a simple specification language for

implicit HCI, based on situational context. We propose an XML-based markup language that supports three different trigger semantics. The language is easily human-readable and also easy to process.

Basic mechanisms of perception to acquire situational context are discussed. In the first example a wearable context-awarenesscomponent built into a tie is described. Also a sensor-based context-awareness device is introduced. Both devices supply context to other devices over a simple request–reply protocol over the serial line.

In a further section, benefits of implicit interaction through situational context to HCI are discussed. In an example, implementation of the feasibility of the concepts introduced earlier is demonstrated.

References

1. Lenat D, The Dimensions of context space. 1999; 1-78. http://www.cyc.com/publications.html.
2. NCR Corp. Mülleimer informiert Supermarkt. I http://www.heise.de/newsticker/data/anm-28.10.99-001/.
3. Maes P, P. Maes on Software Agents: Humanizing the global computer. IEEE Internet Computing July-August. 1997; 10-19
4. Beadle P, Harper B, Maguire G.Q. and Judge J. Location Aware Mobile Computing. Proceedings of IEEE International Conference on telecommunications. Melbourne, Australia, April 1997; 1319-1324
5. Leonhard U, Magee J, Dias P. Location service in mobile computing environments. Computer & Graphics. Special Issue on Mobile Computing. 20, (5), September/October 1996; 627-632
6. Schilit BN, Adams NL, Want R. Context-aware computing applications. Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. IEEE Computer Society. 1994. 1-5. http://www.fxpal.xerox.com/people/schilit/wmc-94-schilit.pdf
7. Cheverst K, Blair G, Davies N, and Friday A. The Support of mobile awareness in collaborative groupware. Personal Technologies 1999: 3; 33-42
8. Pascoe J, Ryan N. S, and Morse DR, Human computer giraffe interaction: HCI in the field, Workshop on Human Computer Interaction with Mobile Devices, University of Glasgow, UK, 21-23 May 1998, GIST Technical Report G98-1. 1998; 56-64.
9. Schmidt A, Beigl M, Gellersen H-W. There is more to context than location. Computers & Graphics Journal, Elsevier, 1999; 893-902.
10. Brown PJ, Bovey JD, Chen X. Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications, October 1997; 58-64
11. Schmidt A, Aidoo KA, Takaluoma A, Tuomela U, Van Laerhoven K, Van de Velde W. Advanced interaction in context. 11th International Symposium on Handheld and Ubiquitous Computing (HUC99), Karlsruhe, Germany, 1999 & Lecture notes in computer science; 1707, Springer, 1999; 89-101
12. Brown, PJ. The stick-e Dokument: a framework for creating context-aware applications. Proceedings EP'96, Palo Alto, CA. (published in EP-odds, 8; 259-72) 1996.
13. Schmidt A, Beigl M, Gellersen H-W. Sensor-based adaptive mobile user interfaces. In Proceedings 8th International Conference on Human-Computer Interaction, München, Germany, August 1999. 2; 251-255
14. Harter A. and Hopper A. A distributed location system for the active office. IEEE Network 1994: 8; 62-70
15. Esprit Project 26900. Technology for enabling Awareness (TEA). 1998. http://tea.starlab.net /.
16. Schmidt A, Forbess J. What GPS doesn't tell you: determining one's context with low-level sensors. The 6th IEEE International Conference on Electronics, Circuits and Systems, September 5 - 8, 1999, Paphos, Cyprus. 1999; 517-525
17. Sawhney N, and Schmandt C. Nomadic Radio: a scaleable and contextual notification for wearable audio messaging. "Proceedings of the CHI 99, Pittsburg, USA 1999; 96-103
18. Schmidt A, Gellersen H-W, Beigl M. A wearable context-awareness component - finally a good reason to wear a tie. In Proceedings of the third International Symposium on Wearable Computers. San Fransico, 18-19. Oct. 1999; 176-177
19. Golding A, Lesh N. Indoor navigation using a diverse set of cheap wearable sensors. In Proceedings of the third International Symposium on Wearable Computers. San Fransico, 18-19. Oct. 1999; 29-36
20. Gellersen H-W, Beigl M, Krull H. The mediacup: awareness technology embedded in an everyday object, 11th International Symposium on Handheld and Ubiquitous Computing (HUC99), Karlsruhe, Germany, 1999; 308-310
21. Farringdon J, Moore AJ, Tilbury N, Church J, Biemond PD. Wearable sensor badge & sensor jacket for context awareness. In Proceedings of the third International Symposium on Wearable Computers. San Francisco, 18-19. Oct. 1999; 197-113
22. Goldstein M, Book R Alsiö G, Tessa S. Non-keyboard QWERTY touch typing: A Portable Input Interface For The Mobile User. Proceedings of the CHI 99, Pittsburg, USA 1999; 32-39

Correspondence to: Albrecht Schmidt, Telecooperation Office (TecO), University of Karlsruhe, Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, Germany. Email: albrecht@teco.edu