Abstract

In this article smart appliances are characterized as devices that are attentive to their environment. We introduce a terminology for situation, sensor data, context, and context-aware applications because it is important to gain a thorough understanding of these concepts to successfully build such artifacts. In the article the relation between a real-world situation and the data read by sensors is discussed; furthermore, an analysis of available sensing technology is given. Then we introduce an architecture that supports the transformation from sensor data to cues then to contexts as a foundation to make context-aware applications. The article suggests a method to build context-aware devices; the method starts from situation analysis, offers a structured way for selection of sensors, and finally suggests steps to determine recognition and abstraction methods. In the final part of the article the question of how this influences the applications is raised and the areas of user interfaces, communication, and proactive application scheduling are identified. We conclude with the description of a case study where a mobile phone was made aware of its environment using different sensors. The profile settings of the phone (ringing mode etc.) are automatically selected according to the real world situation the phone is used in.

How to Build Smart Appliances?

Albrecht Schmidt, Uni Karlsruhe
Kristof Van Laerhoven, Starlab Research

Consumer appliances are often advertised as smart devices since they have enhanced features beyond basic functionality. However, these devices are often regarded as complex and complicated rather than smart. Our notion of a smart appliance is different; we suggest that *smart devices are devices that are not ignorant about their environment and context*.

From our working definition it can be seen that the understanding of context, context awareness, and the concepts that enable context awareness are central to our work. In order to gain this understanding we will discuss the following issues in more detail in this article:

- What is context? How do we relate a certain real-world situation to the more abstract notion of context?
- What is behind the idea of capturing information to approximate context? What technologies are available to capture information about the situation? How can context sensing be implemented?
- What mechanisms and methods are useful to translate the information about the situation captured to a more abstract concept of context?
- How can applications be built to make use of context? How does this help to make human -computer interaction more implicit and therefore more natural?

To answer the questions stated above, we first introduce a terminology for context awareness, then talk about sensing the real world and ways for abstractions, leading to an architecture for context-aware devices. Then we introduce a method to build such devices and discuss possible application domains. We then describe a feasibility study in which we developed a smart mobile phone that is aware of the context of usage and activity of the user. This case study was also used to validate our method.

Up to the mid-'90s, context-aware devices were a synonym for location-aware devices [1, 2], but context means much more than location [3]. We use the term context in a more general way to describe the environment, situation, state, surroundings, task, and so on. In the *Technology for Enabled Awareness* (TEA) project [4] we define context awareness as knowledge about the user's and device's state, including surroundings, situation, and location.

From our experience we learned that it is important to discriminate between the real-world situation, the data captured to represent the situation, the abstract representation of the situation, and the behavior of the application according to this information. In current literature this discrimination is often not clear. We suggest the following terminology:

- Situation or situational context: These terms describe the real-world situation.
- Sensor data or situational data: These terms describe the data that is captured to represent the situation.
- Context or context knowledge: These terms describe the abstract description of the real-world situation.
- Context-aware application or context-sensitive application: These terms refer to applications (software, hardware, appliances) that change their behavior according to the context.

Sensing the Environment

When thinking of appliances that are not ignorant about their context, the important question of how to get this information arises. In this section we outline our approach based on sensing technologies to acquire sensor data on a specific situation that is a prerequisite for the abstraction step.

Sensor Data Is Related to a Situation

Context is an abstract concept and therefore difficult to capture directly. Nevertheless, it is feasible to capture sensor data in a certain real-world situation, as described in [5–7], which can then be used to approximate the context, as described later. The basic assumption of the concept of sensor-based context-awareness is: "*In a certain situation, specific sensor data is frequently similar to sensor data of the same situation at different times*."

Looking at the examples in Table 1 we can see that we can make assumptions on specific sensor values we can get if the user is in a certain situation. When implementing context awareness the idea is to use the sensor data and predict the current situation or even to forecast a situation. The design of a system that is context-aware includes consideration of what types of contexts may be useful to recognize and what sensing

1070-9916/01/$10.00 © 2001 IEEE

Situation	Sensor Data
User sleeps	It is dark, room temperature, silent, type of location is indoors, time is "nighttime", user is horizontal, specific motion pattern, absolute position is stable
User is watching TV	Light level/color is changing, certain audio level (not silent), room temperature, type of location is indoors, user is mainly stationary
User is cycling	Location type is outdoors, user is sitting, specific motion pattern of legs, absolute position is changing.

Table 1. *Real-world situations related to sensor data.*

technology is needed to do this.

When looking at what type of sensors can provide significant information to implement context-aware systems, we have to keep in mind that the sensors just give some information about the situation — but these can be mapped to contexts, using specific methods (e.g., AI, logic, statistics).

Capturing Data

Sensor technology is widely applied in robotics, machine vision, and process control. Advances regarding issues such as size, power consumption, processing requirements, and cost-effective production enable integration in devices and appliances. Since these products tend to compete primarily over price, cost of add-on technology is very critical, and hence we also include simple and affordable technologies in this overview. In this section a brief overview lists some sensors that have been examined in our research.

Light Sensors — Single optical sensors (photodiode, color sensor, IR and UV-sensors, etc.) supply information on the light intensity, density, reflection, color temperature (wavelength), and type of light (sunlight, type of artificial light, etc). We evaluated different light sensors which offered sensitivity for a specific wavelength or for a specific spectrum. Light sensors proved a source of rich information at very low cost since energy consumption and price are very low. Beyond the obvious, interesting information on movement (especially in artificial environments using multiple sensors) can be acquired.

C-MOS Camera — Using cameras, a wide spectrum of information can be sensed, such as visual information about the environment that can be obtained with little processing (e.g., main color, motion) or richer contexts that need more processing power (detection of objects, landmarks, people, gestures etc). Many algorithms are available to gain further information such as a color histogram, recognition of shapes and objects, and motion tracking. Cameras can be cheap, but processing power and storage needs are often large. Also, a certain percentage of users feel uncomfortable being watched by a device.

Audio, Microphones — Microphones can give very interesting information even when using minimal processing. Calculations on a microcontroller with less than 200 bytes of RAM proved to contribute interesting information, such as noise level, type of input (noisy, music, speaking), and base frequency. Multiple microphones can acquired richer information, such as which way the user is holding the device. Using this type of audio analysis is very cheap, while it can be extended up to speech recognition by using more processing power. An interesting aspect is also the use of ultrasonic sensors, to augment human sensory capabilities, for instance.

Accelerometers — Accelerometers can provide rich information to facilitate context awareness. Sensors offer information on the inclination, motion, or acceleration of the device. Typical sensors are mercury switches, angular sensors, and accelerometers. Contexts like orientation or movement of the device, having the device stationary on a table, and driving in a car can be indicated by an accelerometer's data. Acceleration is especially interesting in examination of usage patterns.

Location — Position, location, collocation, and proximity of users, devices, and environment provide important information; see [8, 9] for examples. Outdoors, the Global Positioning System (GPS) is mostly used for fine-grained location sensing, but coarse location information is also available from cellular network infrastructures such as the Global System for Mobile Communications (GSM). Indoors, location sensors are typically embedded in the environment, as in the Active Badge system. Collocation can be sensed with, say, radio beacons.

Touch — Devices that are directly handled by users can benefit from a touch sensor, which can be directly implemented with conductive planes (e.g., skin conductance, human as capacitor), or indirectly using light sensors or temperature sensors. These sensors can reduce energy consumption significantly, especially for devices that only need to be operative in the user's hand.

Temperature — Most temperature sensors are cheap and easy to use. In some applications the information provided is helpful (to detect body heat, for instance, and in arctic or desert environments), but for many other applications the information gain is minimal.

Air Pressure — Air pressure gives an indication of the altitude and also of changes in pressure when used differentially. For some applications this can provide helpful hints about the environment (e.g., a closing door).

Passive IR Sensors (Motion Detector) — In stationary devices, this sensor is of great value, whereas in mobile devices the information is less valuable, since movement of the device itself is detected as well. When selecting these sensors, the fact that they have a directed input (usually an angle between 30° and 180°) must be kept in mind.

Magnetic Field —This sensor offers information similar to a compass, so the direction of a device or movement can be determined. In our experiments we found that in modern environments (e.g., offices with computer monitors) this sensor can give false information. Nevertheless, there are application areas where this can be of significant value.

Gas Sensor — We examined a number of different gas sensors. The main problem for mobile applications is the delay in measurement and the enormous energy consumption. These sensors need to be heated (often around 1 W for about 1 min) before a reading can be taken, and the readings normally differ very little, so its value for general context-aware applications is not obvious. Specific devices for security forces, firefighters, and mining personnel could benefit from this, though.

Biosensors — Many devices are personal, and user awareness can be improved by using biosensors that measure skin resistance, blood pressure, and so on to obtain useful contexts in sports and medical applications. With additional processing,

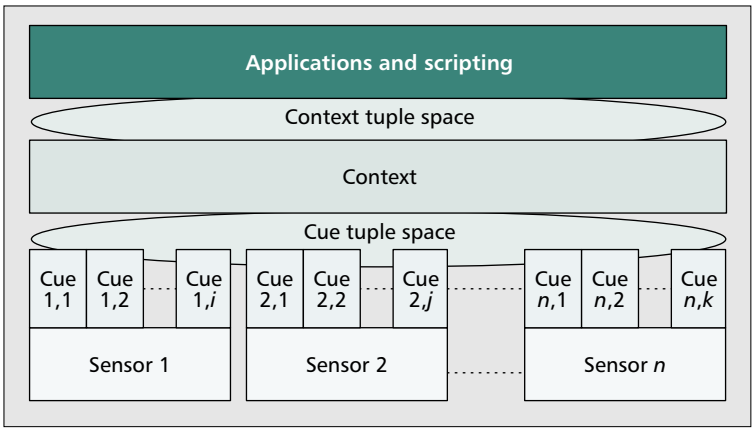

Figure 1. *Layered architecture for context-aware systems.*

awareness of the emotional state of the user may be obtained.

Specialized Physical, Mechanical, and Chemical Sensors — Sensors for touch, temperature, or air pressure may be integrated for use in more specialized applications. For particular mobile work settings, further sensors like gas concentration and radiation sensors may be added to augment the user's sensory capabilities and facilitate automated information capture.

Multiple and Distributed Sensors — In general, multiple simple sensors of the same type (e.g., light sensors) distributed over the device can provide significant, robust information at a cheap cost. Furthermore, putting the sensor in the right place can save a lot of processing power. It is therefore often important to find the most appropriate places to put a sensor; for example, multiple microphones to determine the number of participants in a conversation.

No-Power Sensors — Beyond the sensors listed above, there is a group of sensors that seems especially useful if the system needs to be designed for extremely low power consumption [5]. Sensors like metal ball switches, mercury switches, or solar panels can wake up a sleeping microcontroller while consuming minimal energy in the sleep phase.

A description of possible sensing technologies and more technical details on sensors are available at our project Web page, http://tea.starlab.org/tea_sen.html.

Constraints on Capture

When developing, designing, and building devices that facilitate context awareness, the technical and economic constraints have to be taken into consideration. Technical constraints are related to the type of devices that should be enhanced by context awareness, economic constraints are mainly bound to the cost of the device, and social constraints are based on the anticipated users. For mobile devices the following issues are of major interest; many of them are also important when building embedded or stationary devices.

Portability, Usability, and Design — The introduction of sensing technology should not compromise the portability of the device. Especially size, weight, and physical robustness of sensors are a concern as well as how sensors can be integrated within the whole device.

Power Consumption — Especially for battery-driven mobile devices like mobile phones or PDAs, power consumption is critical. To argue in favor of sacrificing battery power, the added value for the application and the user must be important.

Calibration — Sensors that need specific calibration or have a tendency to decalibrate are difficult to use in consumer electronics. Therefore, post-processing mechanisms to work on relative values are favored.

Setup Time — The time to get a device in a usable mode is a critical parameter for success of devices; therefore, sensors that need a warmup time are only useful for very specific devices.

Reliability — To make context-aware devices successful it is important to build them so that they are reliable. This is on one hand connected to the sensors used, and on the other to methods and algorithms that abstract from sensor data.

Price and Cost — The additional cost introduced by sensors, additional processing power, and development cost has to be related to additional benefit an application gains by being context-aware.

Unobtrusiveness — The appearance of a device should not be changed by the introduction of sensing technology. Especially when devices are equipped with complex optical or acoustic sensing capabilities, users are often concerned about their privacy.

Architecture: From Sensor Data to Applications

To build a flexible yet efficient system, we suggest a layered architecture, similar to that proposed for multisensor location tracking [1]. As depicted in Fig. 1, the architecture consists of four layers: sensors, cues, contexts, and an application layer. With interfaces between the layers, the architecture also caters for the distribution of sensors, cue extraction, context processing, and applications. From our experience it is useful to keep the cue processing close to the sensors, and to distribute the cues and contexts.

When we mention sensors we mean both physical and logical sensors. Physical sensors are electronic hardware components that measure physical parameters in the environment. Information gathered from the host device (current time, GSM cell, etc.) is considered a logical sensor. Each sensor is regarded as a time-dependent function that returns a scalar, vector, or symbolic value.

Depending on the application and on the distribution of the sensing devices, the communication between the layers can be buffered by a tuple space. All cues generated are accumulated in a tuple space. The context layer can then read and use the cues. All contexts that are produced can then be put into another tuple space where the context-aware applications can access them. If the infrastructure is distributed, the tuple space can be used to communicate between the components. To avoid a large amount of tuples in the space, it is useful to give each item a time to live. This should be regarded in the implementation, and items should be removed automatically after they have expired.

Cues

The concept of cues provides an abstraction of physical and logical sensors. As seen from Fig. 1, each cue is dependent on one single sensor; but using the data of one sensor, multiple cues can be calculated.

In building prototypes, the concept of cues proved to be very useful to make changes of the hardware transparent to the context recognition layer. When including new sensors with different characteristics, only changes in the correspond-

ing cues must be adapted. Cues are also a way to reduce the stream of data provided by the sensors.

Cues are one way to reduce the amount of the data provided by the sensors. To implement cues we suggest the usage of statistical functions. They can either provide a summary of the values over time or help to extract features from the raw data that characterize the data over the last period of time. A reasonable time window depends on the sensor and ranges between 100 ms and a few minutes. Cues can be designed to give a relative value and help to eliminate effects from the decalibration of sensors over the device's lifetime. In the following we provide some examples of cues that have been investigated in our projects:

- **Average**. The average of the data items provided by a single sensor over a given time window is calculated. This can, for example, be applied to data from the light, acceleration, temperature, and pressure sensors. For the acceleration sensor this value also gives the one angle of the device to the gravity vector.
- **Standard derivation**. This value can give some indication of how stable a signal is or how much change there is in the signal. The measure is not useful if the signal carries sometimes wrong values, because even a single value can distort the meaning.
- **Quartile distance**. To get an idea of the variation in the signal, this is a robust measure. Sorting the data and calculating the distance between values at one-quarter and three-quarters proves to be more reliable than using the range (e.g., a few faulty values do not wreck the cue).
- **Base frequency**. For some sensors (e.g., light and acceleration) calculating the base frequency of the signal can give some useful information, such as type of lights (flickering), and activities, such as walking (a certain acceleration pattern).
- **First derivative**. The first derivative of the sensor data indicates the change. It is especially helpful to find transitions (e.g., going into the dark).
- **Time domain values**. Especially when working with little processing power (microcontroller), processing the signal in the time domain is interesting. For simple audio processing we used the ratio between zero-crossings and direction changes of the signal, enabling discrimination between music, speech, and noise.

If sensors or sensing devices are spatially distributed, cues are a useful abstraction for communication. Usually the amount of data is very much reduced from the raw sensor signal.

Contexts

A context is a description of the current situation on an abstract level and is derived from available cues. A system working with just a few sensors and a small set of contexts can use simple if-then rules to calculate the current context. A simple example would, for instance, be:

```
If (light is low and acceleration is nonzero) Then context = "Moving in the dark"
```

As a more general approach, the Kohonen Self-Organizing Map (SOM) and its many variants gave good results as clustering algorithms, since they are able to learn new clusters at any time and are known to handle noisy data pretty well. Linking the produced clusters with context labels results in a system that is able to predict the context later on. The topology-preserving property of the SOM makes it very probable that the nearest label will indeed be the right context. Finally, a Markov chain model could serve as a buffer algorithm to check if transitions between contexts are probable. If the transition is not very likely, the system will not change to the new context. Generally, the longer the system is trained, the better the recognition becomes.

During the project we evaluated a number of other AI methods and mechanisms that we used for calculating context from cues. K-Means clustering and its variants are widely used and fast clustering algorithms, but need a specific number of clusters. We also used recorded sensor data from several contexts to train multilayer perceptrons offline and then implement the nonadaptive calculation on a microcontroller. From our experiments we can state that the methods must be selected according to the given hardware constraints (memory, processing power) and anticipated results (fixed, able to learn, slow reaction, etc.).

To feed the contexts that are generated into a data structure such as a the tuple space explained earlier makes the components in the system much more independent and also opens a way to simulate context-aware applications before the sensors have been developed. An alternative architecture for distributed context-aware systems is described in [10].

A Method to Build Aware Devices

Smartness and awareness are a concept that seems very appealing at first sight; however, this concept does not suit all devices we can envision. In this section we concentrate on the process that leads to the selection of devices and describe the making of smart devices. We will refer to objects, devices, and appliances in the process as *artifacts*.

Six Steps to Build a Context-Aware Application

The process can be characterized in the following steps:

Step 1. Identify the Contexts that Matter – Check if Context Matters at All — In a first step the usage of the artifact that should become smarter is analyzed. It can be concluded from the following questions whether the situation matters or not, or if it is probably not worthwhile to make the artifact aware of its context:

- Is the artifact used in changing situations?
- Do the expectations of the user toward the artifact vary with the situation?
- Is the interaction pattern different in various situations?

For all the situations that matter, identify the conditions of the informational, physical, and social environment. Real-world situations that should be treated the same for the artifact are grouped into one context that is named. Considering the real-world situations in a context, a number of informational, physical, and social variables that discriminate the context (e.g., time interval, number of messages, temperature, value, number of people in the vicinity, relationship with people nearby, etc.) are identified.

Step 2. Find the Appropriate Sensors — For the variables identified in step 1, possible sensors are identified, taking the following points into account:

- Accuracy of the sensor in relation to the variable
- The cost to provide the information (for a cost assessment see below)

The resulting selection of sensors should be done such that the sensors cover all variables with sufficient accuracy at minimal cost. For the selection of sensors it is often useful to first select a number of sensors based on the datasheets. Before the integration (step 3), we recommend testing the real sensors in a laboratory environment individually to identify possible problems.

Step 3. Build and Assess a Prototypical Sensing Device — Build, based on the sensors selected, a prototypical sensing device, say, sensors attached to a board in a similar form fac-

tor as the actual device and connected to data storage on a standard computer. Here it is especially interesting to experiment with the positions of the sensors on the device. Then the sensing device is used in the situations that should be detected and data is recorded. In a next step the recorded data is analyzed (e.g., statistics, clustering) to identify whether or not the raw data differs significantly for the different situations. If the data does not differ significantly, different sensors have to be selected (going back to step 2), different contexts have to be identified (going back to step 1), or in some cases it may turn out that it is not feasible to recognize the contexts at all.

Step 4. Determine Recognition and Abstraction Technologies — A set of cues has to be identified that reduces the amount of data but not the knowledge about the situation. Based on the cues, selected as above and applied to the data recorded, an algorithm is selected that recognizes the contexts with maximal certainty and is also suitable for the usage of the artifact (e.g., adaptive vs. nonadaptive, supervised vs. unsupervised learning). Here again the algorithm is selected so that the knowledge about the situation is not reduced by the processing.

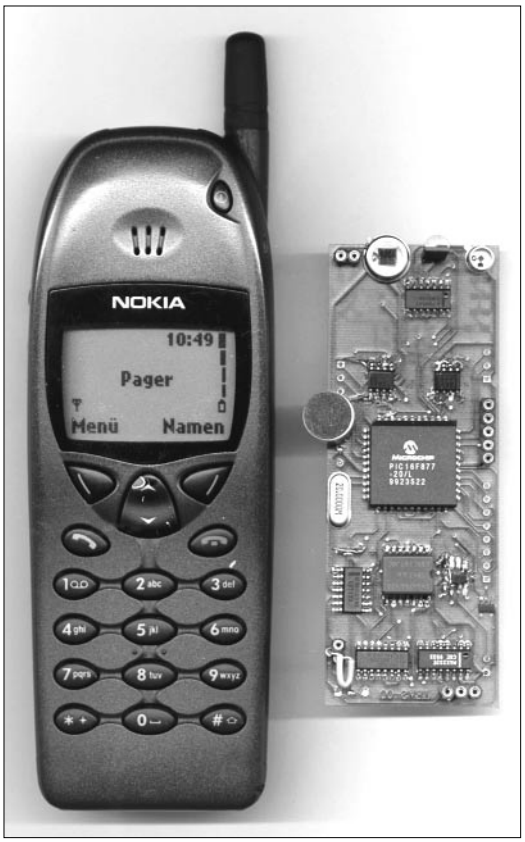

Figure 2. *The sensor board and the enhanced mobile phone prototype.*

Step 5. Integration of Cue Processing and the Context Abstraction — In this step the sensing technology and processing methods are integrated in a prototypical artifact in which the reaction of the artifact is immediate. A design decision can be that the processing is done in the back-end, transparent to the user of the artifact. Using the prototypical artifact, the recognition performance and reliability is assessed in the real-world situations identified. If recognition problems or ambiguities are identified, the algorithms or cues have to be optimized (step 4) or the sensor selection may even need to be rethought (step 2).

Step 6. Build Applications — Build applications on top of the artifact that use the context knowledge. See the next section for a discussion of context-aware applications.

Cost Function

When selecting sensors and algorithms, cost is a major issue. Cost in these terms depends much on the type of artifact (mobile, stationary) that should be built as well as the user group and anticipated retail price. The following issues have to be taken into account, but should be weighted according to the problem.

Power Consumption —If building mobile artifacts, power consumption (sensors and algorithms) is an important factor and should then be a major factor in the cost calculation.

Size and Weight —Again, if designing for mobile devices, the size and weight of sensors and processing power (and, dependent on power consumption, of batteries) are an important issue to look at.

Price of Components —The relation of the price for components (sensors and processing power) to the anticipated retail price compared to the added value provided by context has to be considered.

Robustness and Reliability — If designing consumer appliances, robustness and reliability should be considered, too. Sensors that are highly damageable generally cannot be used.

How Does This Make Applications Smarter?

Applications should behave in accordance with their situations: adapting volume, disabling/enabling loudspeakers, and so on. Mobile users, for instance, may use a phone while going through a wide range of situations: alone in the office, interrupted by a colleague, leaving the office, walking to the parking lot, or driving off in the car. This motivates context awareness to obtain information about usage environments, to be supplied to adaptive applications.

Implicit Human-Computer Interaction

With traditional devices the human-computer interaction (HCI) is explicit. Different modalities have evolved over recent years, such as command-line interfaces, graphical user interfaces, speech interfaces, and gesture interfaces, which have in common that the user explicitly instructs the computer what to do. This concept of explicit interaction contrasts with the vision of invisible or disappearing computing, since a system does not become visible until explicit interaction occurs.

Context can provide a means to improve HCI and move it more toward implicit HCI where the interaction is implicit in the task the user does. Basically, context information can be obtained implicitly through awareness and does not have to be obtained explicitly from the user. Context information can be applied to filter the flow of information from application to user, to address the problem of information overload. Context information can also give additional meaning to the user's input, which combines explicit and implicit HCI.

Three application domains can be distinguished in which we see potential for context awareness:

- *Adaptive user interfaces*, where the utility of interaction styles and display modes depends largely on the surrounding environment
- *Context-aware communication* to filter, reroute, and deliver messages and streams in accordance with a broader communication context, considering issues such as urgency and interruptibility
- *Proactive application scheduling,* supporting the ad hoc style of interaction characteristic for ultra-mobile computing, and their utility for assistance with different tasks in different situations

A Context-Aware Mobile Phone

The context-aware mobile phone illustrates the basic aspects of context awareness. Many mobile phones already have a feature to easily switch from one phone configuration (or *profile*) to another. These settings define how the phone behaves when a call comes in (ringing volume and tone, light alert, automatic answer, etc.). The application layer is therefore already available in the mobile phone, and adding context awareness to it is thus limited to detecting contexts and switching to the corresponding profiles. The mobile phone is also a very obvious mobile application that would benefit greatly from context awareness. It is widespread and well known for disturbing its environment.

The context-aware component was developed according to the method described above in such a way that it would be small enough to fit into the phone's battery pack and could communicate with a Nokia 6110 phone to switch profiles. This enabled extensive user testing, as well as an excellent chance to assess the process of making existing devices context-aware; Fig. 2 shows the sensor board and host phone. The profiles of the mobile phone were selected automatically based on the recognized context. The expected optimal behavior of the phone is more complex; for an in-depth discussion see [4, 6], where issues of external and distributed sensors are also recognized.

For the experiment the following profiles were defined on the phone:

- **Hand**: When the user holds the phone in her/his hand, the audio alarm is not needed. The phone can just alert the user by vibrating.
- **Table**: Here we assume a meeting situation. The phone is almost silent. An incoming call is indicated by a very gentle sound.
- **Pocket**: The phone is set to loud ringing. Here we assume that the phone is put away in a box, suitcase, or pocket and must be silent. The phone still receives calls, so the callers' numbers can be recalled later.
- **Outside**: Here the ring volume goes as high as possible and vibra-alarm is on. All possible ways to get the user's attention are used.
- **General**: General mode is used when none of the above apply.

The user can of course change these settings if, for instance, he or she wants the phone to go silent when put into a pocket. Concerning our experiment we should note that it is feasible under normal usage to recognize these contexts with a certainty of more than 87 percent (or higher, depending on the profile), although it may take up to 30 s until the right profile is detected and switched to.

Conclusion

In our work we consider a smart device one that is not ignorant about its environment. This leads to a wider notion of context, much more related to the word as it is used in everyday language. To provide a common ground for discussion we introduced a terminology that discriminates the real-world situations, the data collected, the abstraction of the data, and the application that makes use of this knowledge. We assume that for a certain situation the data read by sensors is similar to data captured in the same situation previously. Analyzing the sensing technologies available and considering the constraints given for capture, we draw recommendations on sensors.

To build smart devices that make use of the information provided in the environment, we introduced an architecture with a layer for sensors, for cues as an abstraction of a single sensor, the context layer, and a scripting layer providing scripting primitives to the application developer. To develop a system that is aware of its environment we introduce a six-step method.

Finally, we give an overview of application domains, pointing out that context can help to enhance user interfaces, facilitate communication, and provide proactive application scheduling. In a case study we build a mobile phone that is aware of its environment, recognizing if it is in the hand of the user, on the table, put somewhere inside, or outdoors. This awareness is then used to automate selection of ringing modes and answering behavior.

Acknowledgments

This work is founded by the project Technology for Enabling Awareness (TEA) [4], situated in the European Commission's Fourth Framework as part of the call Esprit for IT Mobility. Many thanks to all partners in the project, namely, Kofi Asante Aidoo, Ozan Cakmakci, and Walter Van de Velde at Starlab Nv/Sa, Belgium, Antti Takaluoma and Jani Mäntyjärvi at Nokia Mobile Phones, Finland, and Michael Beigl and Hans-W. Gellersen at TecO, University of Karlsruhe, Germany.

References

[1] U. Leonhardt and J. Magee, "Multi-Sensor Location Tracking," *Proc. 4th ACM/IEEE Int'l Conf. Mobile Comp. Net.*, Dallas, TX, Oct. 1998, pp. 203–14.
[2] B. N. Schilit, N. L. Adams, R. Want, "Context-Aware Computing Applications," *Proc. Wksp. Mobile Comp. Sys. and Apps.*, Santa Cruz, CA, Dec. 1994.
[3] A. Schmidt, M. Beigl, and H. W. Gellersen, "There is More to Context than Location," *Comp. & Graphics J.*, vol. 23, no. 6, Dec. 1999, pp. 893–902.
[4] Esprit Project 26900, TEA; http://tea.starlab.org/, 1998
[5] M. Beigl, H.-W. Gellersen, A. Schmidt, "Mediacups: Experience with Design and Use of Computer-Augmented Everyday Artefacts," *Comp. Net.*, Special Issue on Pervasive Computing, vol. 35, no. 4, Mar. 2001, pp. 401–9.
[6] A. Schmidt *et al.*, "Advanced Interaction in Context," *1st Int'l. Symp. Handheld and Ubiquitous Comp. (HUC '99)*, Karlsruhe, Germany, LNCS 1707, Springer-Verlag, Sept. 1999.
[7] K. Van Laerhoven and O. Cakmakci, "What Shall We Teach Our Pants?" *Proc. 4th Int'l. Symp. Wearable Comp.,* Atlanta, GA, 2000, pp. 77–86.
[8] P. J. Brown, J. D. Bovey, and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace," *IEEE Pers. Commun.*, vol. 4, no. 5, 1997, pp. 58–64.
[9] K. Cheverst *et al.*, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project," *Proc. MobiCom 2000*, Boston, MA, Aug. 2000, pp. 20–31.
[10] D. Salber, A. K. Dey, and G. D. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications," *Proc. Conf. Human Factors in Comp. Sys. (CHI '99)*, Pittsburgh, PA, May 15–20, 1999, pp. 434–41.

Biographies

ALBRECHT SCHMIDT (albrecht.schmidt@acm.org) received an M.Sc. in computing from Manchester Metropolitan University, United Kingdom, in 1996; in 1997 he finished his Master's in computer science at the University of Ulm, Germany. He is a research assistant working toward his Ph.D. at TecO, University of Karlsruhe, Germany. He is especially interested in situated interaction and context awareness in the research field of ubiquitous computing.

KRISTOF VAN LAERHOVEN (kristof@starlab.net) studied computer science at the Limburg University and the University of Brussels, where he was mostly involved in neural networks and robotics at the AI Laboratory. He is a research assistant at Starlab Research Laboratories in Brussels, Belgium, where he is working on adaptive context awareness systems.