Environmental Context Sensing for Usability Evaluation in Mobile HCI by Means of Small Wireless Sensor Networks

Karin A. Hummel and Andrea Hess
Dep. of Distributed and Multimedia Systems
University of Vienna
{karin.hummel|andrea.hess}@univie.ac.at

Thomas Grill
Dep. of Telecooperation
Johannes Kepler University Linz
tom@tk.uni-linz.ac.at

ABSTRACT

In usability evaluations, experiments are often conducted in closed laboratory situations to avoid disturbing influences. Due to non-realistic usage assumptions, this approach has important shortcomings when mobile Human Computer Interactions (m-HCI) have to be evaluated. Field studies allow to perform experiments close to real-world conditions, but potentially introduce influences caused by the environment.

In this paper, we aim at distinguishing application shortcomings from environmental disturbances which both potentially cause decreased user performance. Our approach is based on monitoring environmental conditions during the usability experiment, such as light, acceleration, sound, temperature, and humidity, and relating them to user actions. Therefore, a mobile context-framework has been developed based on a small Wireless Sensor Network (WSN). First results are presented that point at increased delays and error rates of user tasks under induced environmental disturbances. Additionally, we demonstrate the potential of environmental monitoring for understanding user performance.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: Models and Principles—*User/Machine Systems*

General Terms

Mobile HCI, Usability Evaluation, Wireless Sensor Network

1. INTRODUCTION & RELATED WORK

The spreading of mobile devices with increased capabilities in terms of screen and camera resolution, connectivity support ranging from 3G to WLAN and Bluetooth, and positioning support, e.g., via GPS chips on board fosters the deployment of ever new applications and services for mobile devices. These applications are typically used in mobile scenarios but often evaluated in the laboratory to avoid environmental influences.

We focus on the demanding approach to support evaluations carried out *in the field* which allow to provide results more relevant to the applications' real-world usage. Influencing environmental factors can not be eliminated any more and it becomes crucial to differentiate between user performance effects caused by the properties of the mobile application itself or by environmental phenomena. We approach our aim in two steps by monitoring the environmental conditions and relating them to user interactions. In a first step, we actively induce environmental changes to evaluate user interactions under varying – but known – environmental conditions in a laboratory. In a second step, we plan to conduct a field study to exploit the full potential of our approach.

In this paper, we describe the first step and present our new approach using context sensing to capture relevant environmental phenomena for usability evaluation. A context framework is described consisting of wireless networked sensors, which is generic, i.e., the framework may be applied to different use cases of context sensing in mobile scenarios, but customized for mobile usability evaluations.

In the last two decades, many context frameworks have been proposed, like the early works resulting in the context toolkit [13] which provided abstractions to encapsulate sensors using various constructs (e.g., widgets) and the multi-sensor fusion architecture [4].

The Java Context Awareness Framework (JCAF) presented in [2] aims at creating a distributed, loosely coupled, and event-based infrastructure for context-aware applications extensible during runtime by adding context services. Its runtime architecture consists of context service tiers, each handling context information of a particular environment, and context client tiers, which act either as a context monitor or as an actuator that changes context. In [8] the authors describe the Service-Oriented Context-Aware Middleware (SOCAM), which is composed of independent services interacting with each other to support applications that may obtain context information by using push or pull mechanisms. The components of the middleware, including context providers, databases, and locating services, may be distributed over heterogeneous networks.

Recently presented frameworks concentrate on adaptations of context-aware systems during runtime, such as responses to context changes [1] and dynamic reconfigurations [10]. The framework PersonisAD [1] supports personalized ubiquitous services by maintaining models of significant elements of the environment, e.g., people and places. In the Autonomic Context Management System (ACoMS) proposed in [10], fault-tolerant provisioning of context information is achieved by redundant context sources.

In contrast to existing frameworks, our approach is focused on the mobile use of the framework. Hence, one requirement of the framework's system architecture is the ease of distributing sensors on several moving objects and persons. The system is based on a small Wireless Sensor Network (WSN) consisting of small processing units equipped with environmental and motion sensors. The information sink for processing and storing context data is an additional mobile device (e.g., a battery-powered ultra-light notebook).

Mobile usability evaluations have just started to consider influencing environmental context information. In [11], a context-aware patient monitoring application is evaluated by making use of cameras including a tiny camera applied to the mobile device. This additional information is used by pairs of evaluators to discuss each usability problem encountered. In [3], a study examining the effect of changes in motion and light on the user performance is described. Thereby, the test subjects are required to perform tasks, like reading stories and answering multiple choice questions, on a handheld PC as they move within the observation room. An accelerometer is used to measure the movements carried out with the mobile device, whereas the light level in each scenario is recorded in terms of two fixed levels. The measures used to assess the user performance include reading time, response time, and number of correct answers. In [12], the authors present a study investigating the effects of movement on the legibility of text displayed on mobile phones. The set of scenarios involves walking at natural speed and at controlled speed (on a treadmill) while the test subjects read texts taken from newspaper articles as well as random character strings. The influence of walking has been determined by the number of errors, reading velocity, and search velocity in each scenario.

Our approach extends these studies by presenting a context framework based on WSNs including an extended set of sensors for acceleration, light, sound, temperature, and humidity. The software architecture of the framework allows to integrate new sensors very easily and, thus, allows to fully implement the vision of sensing all significant environmental phenomena [14] during usability studies in the field and to overcome the need for restricting environmental influences.

The paper is structured as follows: Section 2 introduces the concept of the context framework. Section 3 gives insights into implementation and calibration issues. Finally, Section 4 describes first experiments using the sensor framework demonstrating the potential for usability studies.

2. CONTEXT FRAMEWORK

For use in mobile scenarios, wireless connections between sensors and a mobile device for context processing have been chosen based on (small) Wireless Sensor Networks (WSNs). The framework has been designed to be generic in the sense that it can be applied in the majority of application fields for mobile context-aware systems.

The main hardware components of the context-sensing system (see Figure 1) are a WSN and a mobile device (e.g., ultra mobile PC or small notebook) acting as a sink. The WSN is composed of a number of distributed nodes which are capable of collecting sensor data on environmental conditions such as light intensity, temperature, humidity, and sound levels as well as an object's acceleration. The context-sensing application running on the notebook processes the data from the WSN and provides a user interface.

The communication between the sensor network and the notebook is enabled through a gateway. The sensor nodes transmit the gathered sensor readings in data packets using ZigBee (IEEE 802.15.4) to the gateway. The gateway is responsible for forwarding the data packets it receives from the network nodes to the notebook and is connected to the notebook via a serial RS-232 interface or a USB link. The particular system setup used is based on Crossbow MicaZ motes [5] responsible for communicating with one another and attached sensor boards for monitoring different environmental phenomena. The distributed placement of sensors allows to apply each sensor where it is most useful.

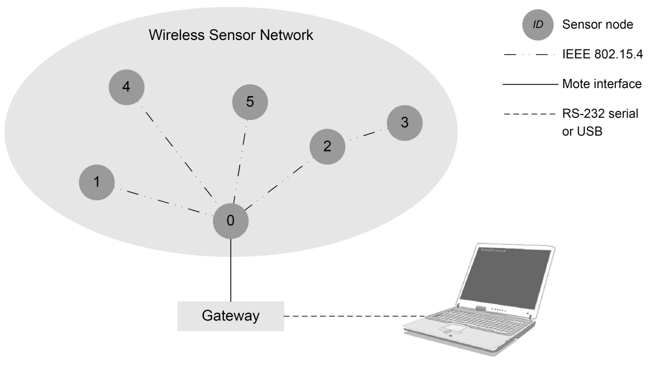

Figure 1: Overview of the context framework's hardware components.

The software on the motes implements basically a star topology. Every sensor node sends its data packets containing sensor readings to the base node (node ID 0 in Figure 1), which is programmed to transfer all packets to the notebook via the gateway. The other nodes are programmed to sample readings from the sensors incorporated into the attached sensor boards and to transfer the readings in configurable intervals. In order to reduce the processing load on the gateway, we adapted the topology in such a way that one acceleration node (node ID 2 in Figure 1) is responsible for aggregating the data collected by other acceleration nodes within the network (in our case, one additional node) and sending the aggregated information to the base node.

Two types of sensor boards are used, namely, Crossbow's MTS310 and MDA300 [6]. The MTS310 sensor board incorporates a thermistor, a photo resistor, a two-axis accelerometer, a magnetometer, a microphone, and a sounder. The MDA300 sensor board provides pre-calibrated temperature and humidity sensors, while the sensors on the MTS310 board need to be calibrated before they can be used for measurements. Table 1 gives an overview of the sensor boards and their sensors for each node used in the prototypical implementation including sampling intervals.

3. IMPLEMENTATION ISSUES

The implementation of the WSN software is based on the TinyOS operating system [9]. The software on each WSN node is composed of the necessary components of TinyOS and application-specific program code. The applications are implemented in nesC [7], a C-based programming language developed to enable the execution of autonomous software

ID	Node	Sensor board	Sampling interval
0	base	connected to gateway	
1	light	MTS310	500ms
2	x/y accel.	MTS310	100ms
3	z accel.	MTS310	100ms
4	temp./hum.	MDA300	4s
5	sound	MTS310	100ms

Table 1: Sensor nodes of the prototype WSN.

programs on motes constrained by memory and energy limitations. An important objective of nesC is the avoidance of runtime errors by analyzing the whole program at compile-time and by eliminating potential sources of bugs such as dynamic memory allocation or dynamic dispatch. Thus, all resources are known statically at compile-time.

The context-processing application running on the notebook which serves as a sink in the WSN is implemented in C#. This application is responsible for the processing of context data and thus, carries out the calibration of sensor readings. If needed, the calibration has been carried out by adapting the readings to additional pre-calibrated reference sensors or, for acceleration, to stationary, stable conditions (no movement). Additionally, the sensors require a start-up phase to stabilize (as a result of a number of experiments investigating the initial phase of data collection, we configured the start-up phase to last for 20 minutes).

Light

The calibration of light readings is realized by mapping the sensor results into units of measurements. The calibration method proposed comprises two mapping equations, namely, a linear function for lower value ranges and an exponential function for upper value ranges. The raw ADC (analog-to-digital converter) readings are converted directly into the final luminance value in lux. The calibration method for light is incorporated into the conversion algorithm of the context-processing application as shown in the algorithm depicted in Figure 2 (where the current configuration settings are given as follows: $a = 0.0893$, $b = 0.0408$, $c = 0.0016$, $d = 0.0146$, and $threshold = 700$).

```
Input: integer lightADC
Output: double luminance
IF (lightADC <= threshold)
  THEN luminance = a * lightADC + b
ELSE luminance = c * e^(d * lightADC)
```

Figure 2: Calibration of light sensor readings.

Temperature and Humidity

The calibration experiments for temperature and humidity showed that it is not necessary to calibrate the readings collected by the MDA300 sensor board since the deviation from our reference measurement instrument is within an acceptable accuracy. Hence, temperature in degrees Celsius and relative humidity are derived from the ADC values with the two conversion formulas suggested by the manufacturer.

Sound

The sound readings are not calibrated by default due to the characteristics of the microphone. The microphone is intended to detect tones at a frequency of 4 kHz (as generated by the sounder of the sensor board). Thus, tones at other frequencies were detected in a reduced intensity. Although the sensed ADC values could not be converted into decibels, the ADC readings can be used as an indicator for sound level changes in the environment as has been evaluated against a calibrated sound meter.

Acceleration

Acceleration is calculated using three dimensions. Since each accelerometer measures the acceleration for only two directions, we arrange two sensor boards at right angle between each other to retrieve x/y/z-axis readings. Finally a force vector is derived from the acceleration values for all three directions.

For calibration, the sensors are not moved and are calibrated in a way to result in zero g. The acceleration motes have to be placed on a horizontal surface because a tilt of the accelerometers would also influence the readings. To perform this 'zero value' calibration, the corresponding value for each axis is derived from the mean value of the readings collected in the last minute of the start-up phase. (Experiments showed that a time period of one minute is adequate for the calculation of the mean value. However, the length of this period can be configured by the user.)[1]

4. MOBILE HCI EXPERIMENTS

Experiments have been conducted to show whether environmental influences cause changes in the user's performance and whether monitoring may give additional insights into a usability experiment. Simultaneously, the experiments demonstrated the robustness and usefulness of the context framework itself for mobile HCI experiments.

To approach the investigation aims, we first have to quantify user actions. Test users have to perform a sequence of predefined tasks by interacting with an application running on a mobile smart phone. Based on this task sequence, *reference timestamps* defining the optimal intended timestamps for user interactions can be defined. The actual user interactions are logged with the actual timestamps by the instrumented mobile application. These logs allow to relate user interactions with the additionally recorded environmental context.

To quantify the user's performance, two metrics are introduced making use of the recorded user logs:

Delay. The delay D is measured in seconds and defined as $D = T_A - T_R$ (where T_A is the actual timestamp of the user activity and T_R is the reference timestamp). In case D evaluates to a negative number, the user acted faster than the expected reference activity is scheduled. The delay is defined only for tasks solved.

Error rate. The error rate E is defined as the number of tasks exhibiting at least one erroneous interaction user log divided by all considered tasks. (E may be calculated for different time slots of an experiment.)

[1]For the mobile HCI experiments described in Section 4, we had to perform one acceleration calibration before starting the experiments.

4.1 Hardware Setup

The user interacted with a mobile smart phone (Qtek S200), while the sensors were applied where they were expected to be most useful for the mobile usability evaluation. To track the person's movements, the acceleration sensors were applied to the left arm of a person (right arm in case the person is left-handed). The sound sensor was placed in the proximity of the test person's ear, while the light sensor was placed near the smart phone to detect light shining onto the display. Finally, temperature and humidity sensors were applied to the right arm (left, for left-handed persons). Figure 3 shows a test person and the used wireless networked sensors. For reproducibility purpose, all scenes were recorded by a camera.

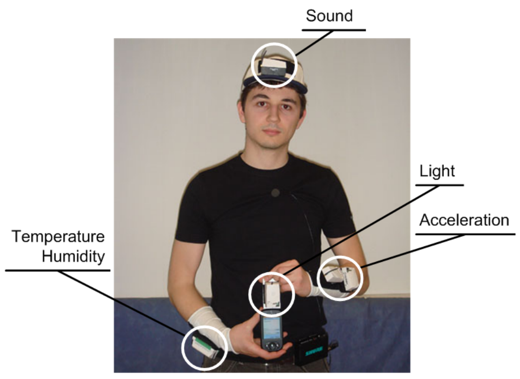

Figure 3: A participant equipped with sensor nodes.

4.2 Usability Study Setup

For first evaluations, we actively changed environmental conditions to be able to observe the user interactions under similar conditions for all test persons and experimental runs. As a consequence, we conducted the experiments indoors in a laboratory-like environment.[2] We explicitly invoked environmental phenomena according to an *environmental time schedule* lasting for five minutes. While environmental conditions were changed, the user interacted with the mobile smart phone according to tasks given by a continuous video projection in the room.

We conducted three experimental runs:

Sitting. In this stationary experimental run, the user is sitting on a table performing the given tasks.

Moving 1 and 2. These experimental runs differ only in the kind of tasks to be solved (as given by different video projections). In both cases, the user is forced to change place while environmental conditions are changed according to the strict environmental time schedule. The schedule consists of subsequent time slots of 30s dedicated to: extensive moving (*Acceleration*), increased light level induced by an reflector (*Light*), increased sound level by playing different noisy sound files (*Sound*), and decreased temperature and changed humidity by opening a window nearby (*Temperature/Humidity*). Times where no environmental changes are induced are titled *w/o*.

[2]The mobile context-framework has also been used in outdoor experiments and proven to be usable in these experiments.

4.3 Results

The following results have been derived from aggregating the performance of seven test persons each performing three experimental runs. The context framework itself worked correctly and stored about 45 MB of sensor data.

Figures 4 and 5 show the aggregated performance results for all seven participants for the time slots corresponding to the different environmental changes in experiment runs *Moving 1* and *Moving 2* – aggregated to *Moving*. (The time slots on the x-axis have been named according to the environmental changes.) The *Sitting Av.* value represents the average performance of users interacting with the mobile application during the whole experiment where environmental conditions have not been changed (scenario *Sitting*).

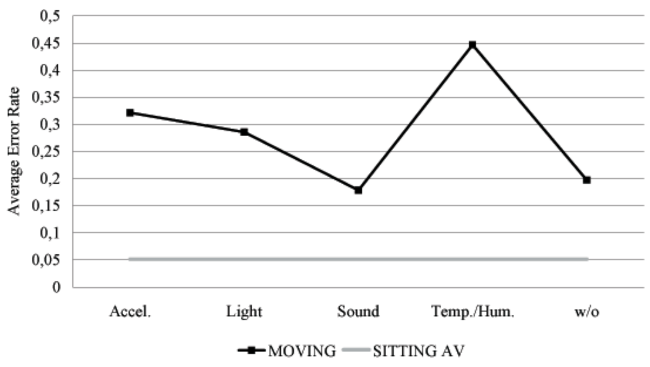

Figure 4: Average error rate in the moving scenarios compared to the mean average error rate in the sitting scenario over all seven participants.

Figure 4 (detailing the average error rate) and Figure 5 (detailing the average delay for the different induced environmental changes) show that environmental disturbances indeed lead to decreased user performance. The error rates are higher ranging from about 0.18 to 0.45 than the average error rate of about 0.05 in the sitting scenario. Similarly, the delays are higher ranging from 3.34s to 7.02s than the average "delay" of about -0.92s in the sitting use case.

Although *Acceleration*-only is differentiated explicitly, the test persons were also slightly moving in the other time slots including *Temperature/Humidity* and *w/o* which explains the unexpected high values for these time slots. It further has to be said that the variance of user performances both for the error rate and the delay were high (e.g., the error rate ranged from 0 to 1 for the test persons in some periods of similar environmental disturbances). Thus, with just this sample of seven persons it is not possible to derive a ranking of most disturbing environmental factors and the results have to be considered to be preliminary.

Figure 6 shows an example how the sensor values can be used to reason about the causes of a user's performance. A 30 second extract (second 20 to 50) of a person's sensor trace during one moving experiment has been chosen depicting two errors and one significant delay of two seconds. The usability evaluator (or evaluation tool) may now use this information – here graphically – to see that for both delay and errors given no noise, light, or temperature/humidity changes in the environment happened, but an increased movement

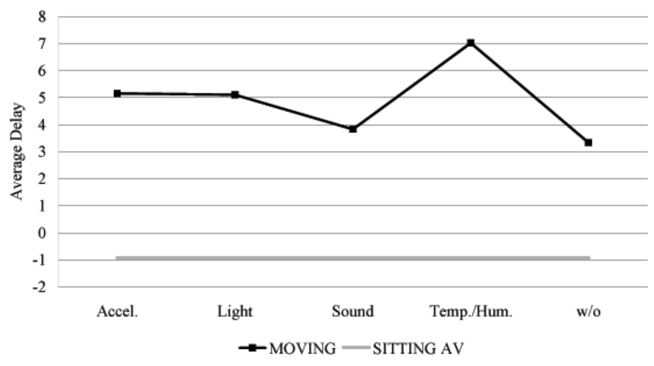

Figure 5: Average delay (in seconds) in the moving scenarios compared to the mean average delay in the sitting scenario over all seven participants.

activity. Automated usability evaluation tools may now classify these two errors and the delay as possibly caused (or at least influenced) by movement. It is up to future work to present methods to derive environmental patterns which are likely to decrease user performance.

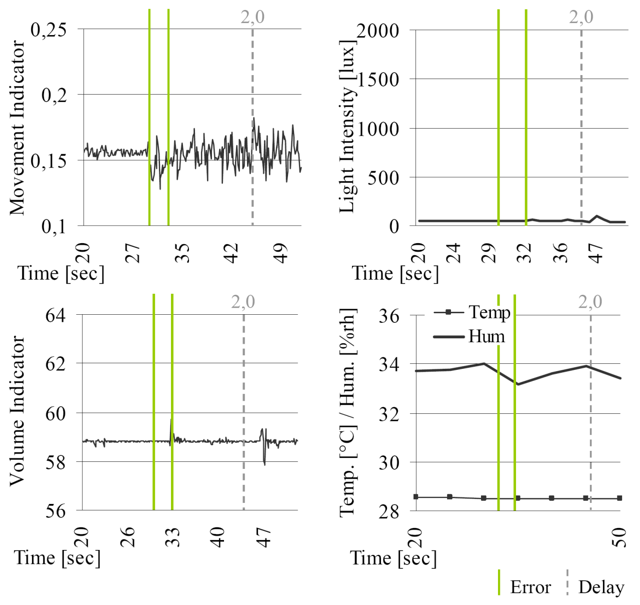

Figure 6: Example demonstrating how to use sensor values to point at possible reasons for errors and delays (delays given in seconds).

5. CONCLUSIONS AND FUTURE WORK

We have presented a mobile context-framework based on small Wireless Sensor Networks (WSNs) and its application to mobile usability evaluation. The main benefits of the presented solution are its support for mobile environments where sensors may be applied easily to different objects (or at different places) and its modular software architecture including a calibration module.

User experiments have been conducted in a laboratory with seven test persons where we changed different environmental conditions. First results showed that the performance of the test persons on the average decreased in terms of higher error rates and delays while performing given tasks. We further demonstrated, how the gathered sensor data can be used to reason about causes of delays and errors. In future work we plan to extend these preliminary results by comprehensive field study experiments and by deriving environmental patterns that are likely disturbing a user.

6. REFERENCES

[1] M. Assad, D. Carmichael, J.Kay, and B. Kummerfeld. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In *Pervasive Computing 2007*, pages 55–72, 2007.

[2] J. Bardram. The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In *Pervasive Computing 2005*, pages 98–115, 2005.

[3] L. Barnard, J. Yi, J. Jacko, and A. Sears. Capturing the Effects of Context on Human Performance in Mobile Computing Systems. *Personal and Ubiquitous Computing*, 11(2):81–96, 2007.

[4] D. Chen, A. Schmidt, and H.-W. Gellersen. An Architecture for Multi-Sensor Fusion in Mobile Environments. In *Int. Conf. on Information Fusion*, pages 861–868, 1999.

[5] Crossbow Technology Incorporated. *MPR-MIB Users Manual*, June 2006.

[6] Crossbow Technology Incorporated. *MTS/MDA Sensor Board Users Manual*, June 2006.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation*, pages 1–11, 2003.

[8] T. Gu, H. Pung, and D. Zhang. A Service-Oriented Middleware for Building Context-Aware Services. *Journal of Network and Computer Applications*, 28(1):1–18, 2005.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. *ACM SIGPLAN Notices*, 35(11):93–104, 2000.

[10] P. Hu, J. Indulska, and R. Robinson. An Autonomic Context Management System for Pervasive Computing. In *Pervasive 2008*, pages 213–223, 2008.

[11] J. Kjeldskov and M. Skov. Exploring Context-Awareness for Ubiquitous Computing in the Healthcare Domain. *Personal Ubiquitous Computing*, 11(7):549–562, 2007.

[12] T. Mustonen, M. Olkkonen, and J. Hakkinen. Examining Mobile Phone Text Legibility While Walking. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1243–1246, 2004.

[13] D. Salber, A. Dey, and G. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *SIGCHI Conf. on Human Factors in Computing Systems*, pages 434–441, 1999.

[14] B. Thurnher, T. Grill, K. Hummel, and R. Weigl. Exploiting Context-Awareness for Usability Evaluation in Mobile HCI. In *uDAY IV*, pages 109–113. Pabst Science Publisher, 2006.