On Context-Sensitive Usability Evaluation in Mobile HCI

Karin Anna Hummel
Department of Distributed and Multimedia Systems, University of Vienna
Lenaugasse 2/8, 1080 Vienna, Austria
karin.hummel@univie.ac.at

Thomas Grill
Department of Telecooperation, Johannes Kepler University Linz
Altenberger Strasse 69, 4040 Linz, Austria
tom@tk.uni-linz.ac.at

Andrea Hess
Department of Distributed and Multimedia Systems, University of Vienna
Lenaugasse 2/8, 1080 Vienna, Austria
andrea.hess@univie.ac.at

Received (received date)
Revised (revised date)

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-HCIs) have to be evaluated. On the other hand, field studies allow to perform experiments close to real-world conditions, but potentially introduce influences caused by the environment, which have not been fully investigated so far.

With this work, we contribute to 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 Wireless Sensor Network (WSN) carried together with a mobile PC. We present results of a small study (seven persons) in the lab which pointed at increased delays and error rates of user interactions under induced environmental disturbances. Hereby, we demonstrate the potential of environmental monitoring for understanding user performance. Additionally, we present novel results of a usability study carried out in the field where we tested 19 persons under varying environmental conditions. The results showed that error rate and delay are influenced by environmental parameters, but in a more complex way than expected a-priori.

Keywords: Mobile HCI, Field Usability Evaluation, Context-Sensitivity, Wireless Sensor Network
Communicated by: to be filled by the Editorial

1 Introduction

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 new rich media applications and services for mobile devices. These applications are typically used in mobile scenarios, but often evaluated in the laboratory (in-vitro) to avoid environmental influences during the

evaluation of the mobile applications.

Although restricting the influence of environmental factors is a valid approach, usability study results suffer from this unrealistic setting and results might mislead mobile application development. Hence, we focus on the demanding approach to support evaluations carried out *in the field* (in-situ) to provide more significant results regarding the applications' real-world usage. Influencing environmental factors – the context – can not be ignored any more and it becomes crucial to differentiate between user performance effects caused by the properties of the mobile application itself, and by environmental phenomena. Additionally, it is also not clear if and how users will react to malicious environmental conditions like increased sun light which worsens the visibility of the mobile device's display.

This paper contributes to context-sensitive usability studies by proposing a solution how to measure and relate context data and user interactions based on a field study. We approach our aim by developing a context-framework for usability studies first introduced in [17] and conduct two different evaluation types. In the first step, we actively induce environmental changes to evaluate user actions under varying – but known – environmental conditions in a laboratory. In the second step, we conduct a field study where real-world environmental conditions were observed. Hereby, we consider two controversial hypotheses:

> *When starting under normal and convenient environmental conditions, the user is likely disturbed by environmental conditions (e.g., light, sound, significant changes of temperature and humidity) and increased movement. These disturbances result in decreased user performance.*

> *When starting under normal and convenient environmental conditions, the user will perceive environmental changes and likely adapt, e.g., by compensating the disturbing factors and increasing the concentration on the task, leading not to decreased user performance – possibly even to better user performance.*

In the remainder of the paper, we describe our approach using context sensing to capture relevant environmental phenomena for usability evaluation. The relation to existing, related approaches is discussed in Section 2. Besides the general extension of state-of-the-art mobile usability studies, the context framework itself is a contribution to the field of mobile context-sensitive computing and designed for easy integration of additional sensors. In Section 3, we describe the design and implementation of this context framework and give further insights into usage, startup phase, and calibration issues. In Section 4, we show how we set up usability evaluations that applied the context framework, introduce the scenarios used, and describe the metrics, i.e., user interaction error rate and delay, introduced to quantify user performance.

In the Sections 5 and 6, we describe the usability evaluations conducted. First, we did a laboratory evaluation with a small user group (seven test persons). Second, a field evaluation was conducted where we tested 19 persons at the university campus of the University of Vienna. In the laboratory experiments, we induced artificial environmental changes to study user reactions under controlled conditions and to validate the framework. The results of this small study showed the potential of the context framework and the approach in general (Section 5). In Section 6, we describe the field study and investigate the environmental

influences with respect to user interaction errors and user interaction delays. Finally, we conclude our work in Section 7 and give an outlook on future work and open research issues.

2 Related Work

During the last decade, a number of frameworks for sensing and using environmental data have been proposed for different purposes, like location-aware services, home automation, and environmental monitoring. Context-awareness has been stated to be of major importance for user satisfaction with, e.g., situated applications like tourist guides [8, 14]. Most of the existing frameworks use an on board or stationary system to collect, store, and use context data. Since our goal is to collect context data during usability evaluations for mobile applications in the field where the context might influence users most, this restriction is not satisfactory.

To relate our work to state-of-the-art approaches, we, first, describe recent mobile usability evaluation research and, second, context frameworks. Hereby, we further discuss to which extent existing context frameworks fulfill our requirements for context-sensitive usability evaluation for mobile HCI.

2.1 State-of-the-Art of Mobile Usability Evaluation

Mobile usability evaluations have just started to consider influencing environmental context information. In [19], 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 [4], 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 using 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 [20], 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.

The influence of mobility on the type of evaluation is analyzed in [22]. The authors confirm a positive influence of applying evaluations in the field compared to laboratory evaluations done for mobile applications. They found out that in laboratory settings the error rate is slightly higher than in the field and relate this issue to the influences of the formal test environment. The analysis of their results concentrates on task evaluation and neglects the influence of context to the results of their analysis.

In [11], other findings are reported. The authors compare the quality of evaluations made in the laboratory and in the field based on the problems they found. The findings include reasoning that more problems and in particular more critical problems are identified in the field. Additionally, the performance of test users was worse in the field which is based on the effect of not being able to concentrate well enough on the tasks due to environmental

disturbances and introduced by the social situation they were exposed to.

Advantages and disadvantages of mobile usability evaluations in the field and the laboratory are also elaborated in [21]. The authors state that the advantage of laboratory evaluations lies in the clearness of the test scenarios and the established way of analyzing the results of the evaluations due to consistent evaluation settings. However, during the analysis of the field evaluations they have discovered that the metadata about the evaluation provides a much deeper insight to usability problems and helps to detect issues that did not even appear during an evaluation conducted in the laboratory setting.

All of the studies conducted have identified a positive effect on the outcome of the evaluation when conducting a field evaluation. In spite of this positive effect of field studies, Kjeldskov and Graham state in their survey of mobile HCI research methods [18] that about 71 percent of all conducted usability evaluations are in-vitro studies. The additional costs of studies conducted in the field are a major cause of this situation.

2.2 Context Frameworks

In this section, we will review selected representative approaches of context frameworks. Two of the first frameworks proposed for context sensing are the context toolkit [24] which provides abstractions to encapsulate sensors using various constructs (e.g., widgets) and the multi-sensor fusion architecture [7]. The multi-sensor fusion architecture is a layered awareness system fusing context information to describe the environment of mobile devices. Each layer of the architecture aggregates the context information provided by the underlying layer on different levels, namely, data, feature, and decision levels. Requirements and an architectural approach for practical development of context-sensitive mobile applications are presented and summarized, e.g., in [1, 6].

The Java Context Awareness Framework (JCAF) presented in [3] 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 [13], 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 and 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 [2] and dynamic reconfigurations [16]. The framework PersonisAD [2] 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 [16], fault-tolerant provisioning of context information is achieved by redundant context sources.

To the best of our knowledge, none of the general purpose context frameworks proposed in literature so far have been applied to context-sensitive usability evaluations to take context as an influencing factor into account. In [5], a context-aware system for wearable computing used in a usability study that investigates whether the use of context-awareness improves

user performance is described. Here, an information retrieval application provides location-responsive information based on GPS data collected by a wearable computer. Conversely to our work, location is thereby not considered as a factor influencing the user itself. It is rather studied whether integrating context-based services into applications improves their usability. One attribute of the framework equivalent to the framework proposed in this work is that it has been developed for the use in mobile scenarios.

In the field of mobile HCI, three types of context data provisioning have been applied to usability studies in the past, but without making use of a context framework. The context information has either been (i) captured by an observing person, has been (ii) provided by a controlled environment, or has been (iii) generated by means of simulation. When context information is captured by observers, handwritten or manually entered notes are taken containing information on changed context parameters and the approximate point in time these changes occurred. In case of a controlled environment, the values of context parameters (e.g., the exact light intensity or sound level) are pre-defined for each time interval of a test scenario. Examples of usability evaluations with controlled context parameters can be found in [4, 20]. In addition, context information is sometimes gathered from a simulated environment. In [23] the authors present an empirical study investigating the effect of application type and location context on mobile usability. Instead of capturing context information, the provisioning of location context via GPS has been simulated while the study was conducted in a closed environment. These methods for context data generation are not fully satisfying when conducting mobile HCI studies as we address since they cannot overcome the restriction of an unrealistic usage setting.

In contrast to most of the existing context frameworks, our approach is focused on the mobile use of the framework. One requirement of the framework's system architecture is ease the application of sensors on several moving objects and persons and to avoid (data) wires. The design is therefore based on a small Wireless Sensor Network (WSN) consisting of small processing units equipped with environmental and motion sensors which can be applied to moving objects and persons. Additionally, the overall architecture includes an information sink for processing and storing context data (e.g., a battery-powered ultra-light notebook).

3 Context Framework

For use in mobile scenarios, wirelessly connected sensors and a mobile device for context data processing are chosen based on (small) Wireless Sensor Networks (WSNs). The framework is 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 or person's acceleration. The context-sensing application running on the notebook processes and logs the context 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 over a ZigBee (IEEE 802.15.4) low-energy wireless network to the gateway. The gateway is responsible for

forwarding the data packets it receives from the sensor nodes to the notebook and is connected to the notebook via a serial RS-232 interface or a USB link. The particular prototype system setup used is based on Crossbow MicaZ motes [9] responsible for communicating with one another and attached sensor boards for monitoring different environmental phenomena. The distributed nature of the sensors allows to apply each sensor where it is most useful.

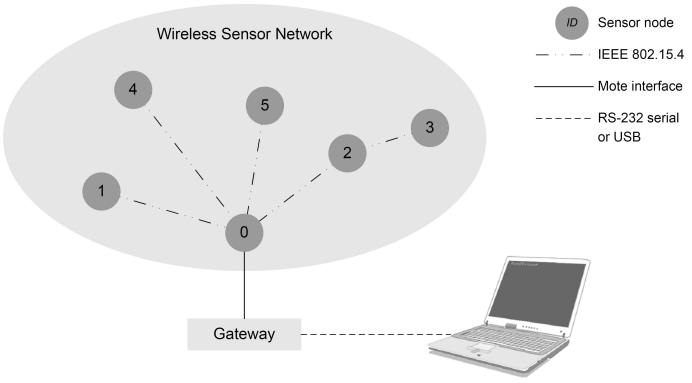

Fig. 1. Overview of the context framework's hardware components.

The software running 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 time 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.

Table 1. Sensor nodes of the prototype WSN.

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

Two types of sensor boards are used, namely, Crossbow's MTS310 and MDA300 [10]. 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 precalibrated 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 the used sampling intervals.

3.1 Implementation Issues

The implementation of the WSN software is based on the TinyOS operating system [15]. 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 [12], a C-based programming language developed to enable the execution of autonomous software 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 dispatching. 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 required by the sensor hardware, the calibration is 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). We now describe selected calibration details of interest.

Light

The calibration of light readings is realized by mapping the sensor results into units of measurements (lux). 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)
```

Fig. 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 as provided in the source code delivered with the sensors 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. (We use the unit *sound level* to reflect to the values extracted from this sensor.)

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 value (in g) 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.)[a]

4 Usability Study Setup and Evaluation

To demonstrate the benefits of our approach and to validate the context framework and its usefulness, we conducted two studies, one in a laboratory setting and the other in a field setting. The aim of the studies was to track environmental context data, movement, and user performance with respect to a given task and to relate these values with one another.

In both studies, the study participants were tested under *three similar scenarios* determining movement and context changes. These scenarios can be thought of as a script which is communicated to the test users by the evaluator. Additionally, the *same mobile application tasks* (user interactions) are performed in both studies. To be able to compare user performance quantitatively, the *error rate* and *delay* of user interactions are defined and, again, used both in the laboratory and the field study.

4.1 User Tasks

The usability laboratory and field studies are based on the same tasks. Users are instructed by video messages (in the lab) and audio messages (in the field) which define when to start and fulfill a particular task. Figure 3 shows a typical instruction given to the test user. In the field study, the instructions are given to the user by means of an audio representation of the instruction via an MP3 player. It was assumed, that in the field video presentation would be difficult to manage and additionally distract the users.

Tasks are performed by using a mobile example application termed *MobEval*, a note-taking

[a]For the mobile HCI experiments described in Sections 5 and 6, we had to perform an acceleration calibration before starting the experiments.

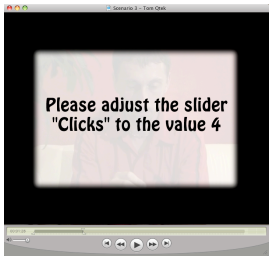

Fig. 3. Instruction video.

tool for mobile usability evaluations. The tool captures user interactions which are evaluated after the experiment. Figure 4 shows a screen shot of the user interface of the MobEval tool and Figure 5 shows a typical logfile generated by the mobile tool. The user interactions have been further classified according to well-known interaction patterns resulting in four interaction types, they are: entering text notes (type 1), manipulating a slider / continuous status event (type 2), entering values by radio buttons / discrete status event (type 3), and clicking at some button / toggle event (type 4).

Fig. 4. Screenshot of the MobEval tool.

Fig. 5. MobEval logfile.

The users are instructed to move according to three scenarios, that are, (*i*) a stationary scenario where test persons are sitting and hardly any environmental changes happen (4.5 minutes), (*ii*) a shorter movement scenario (4.5 minutes), and (*iii*) a longer movement scenario (6.3 minutes). In the laboratory setting, the evaluator tells the participant where to move to while in the field the participant simply follows the instructor leading to context changes. In the laboratory setting, artificial changes of single environmental properties (e.g., light) are induced following a strict plan for each mobile scenario.

Users are prepared and instructed of the overall study procedure in advance. Beforehand he/she has some time to get used to the particular evaluation setting and is equipped with the sensors and the smartphone. The evaluator explains exactly what will happen in order

to reduce nervousness caused by the evaluation situation. The user is also instructed to formulate problems upon occurrence during the experiment (thinking aloud method). These comments are noted by the evaluator.

4.2 Hardware Setup of the Evaluation

The users interacted with a mobile smartphone (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 smartphone to detect light shining onto the display. Finally, temperature and humidity sensors were applied to the right arm (left for left-handed persons). Figure 6 shows a test person and the used wirelessly networked sensors. For reproducibility purpose, all scenes were recorded by a camera. Additionally, we used an external audio logger in the field to get sound values from a different device and an audio player playing the audio instructions.

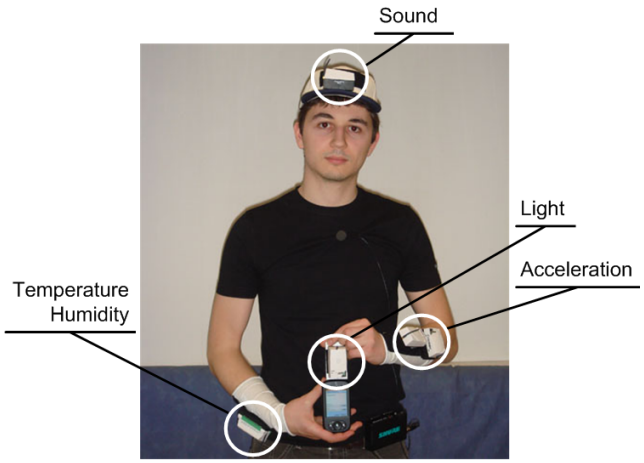

Fig. 6. A participant equipped with sensor nodes.

4.3 Logging and Measures

User interactions are logged as well as context data. The logfile produced by the mobile tool MobEval is based on XML (see Figure 5), which simply allows to transform the logs to any required format and to analyze the logs with commonly used tools like Microsoft Excel or the statistic tools R and SPSS. Hereby, user interactions are logged together with a timestamp. Additionally, raw context data are logged together with a timestamp by the context framework. By synchronizing the internal clocks of the smartphone executing the MobEval application and the context framework mobile PC, the logs can easily be related.

In more detail, user interactions take place within a time interval (time frame), hence, we define a time frame in the range of a few up to 10 seconds for each interaction (type). Therefore, we calculate a *time frame lower* and a *time frame upper* for each timestamp

assigned to a particular user interaction. Sensor values are considered to be of relevance for an operation, if they have occurred within the particular time frame limited by time frame lower and time frame upper.

To compare user interactions under different environmental conditions, we quantify user actions. Therefore, we describe an ideal task sequence and ideal *reference timestamps* defining the optimal intended timestamps for user interactions.[b] As the actual user interactions are logged with the actual timestamps by the instrumented mobile application as well, it is possible to relate actual and reference timestamp. The detailed metrics used to quantify the user's performance are:

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-free interactions).

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.)

In our evaluations, we analyze each sensor type independently of other types and assume that delay and error are independent of past delays and errors. This assumption is expected to hold for most of the user interactions because pauses are planned in the schedule leaving time to recover after an error or long delay.

5 Mobile HCI Experiments in the Lab

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.

For first evaluations, we actively changed environmental conditions to be able to observe 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. We explicitly invoked environmental phenomena according to an *environmental time schedule* lasting for about 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 length and kind of tasks to be solved (as given by different video projections). In both cases, the user is forced to

[b] The reference timestamps are defined based on experiences gathered in past user studies with similar interaction types.

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*.

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.

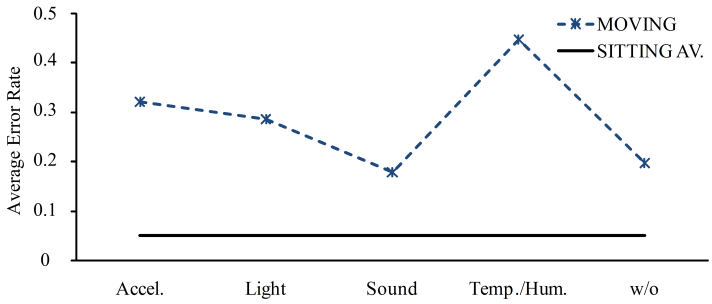

Fig. 7. Average error rate in the moving scenarios compared to the mean average error rate in the sitting scenario over all seven participants.

Figures 7 and 8 show the average 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 *Sitting* scenario.

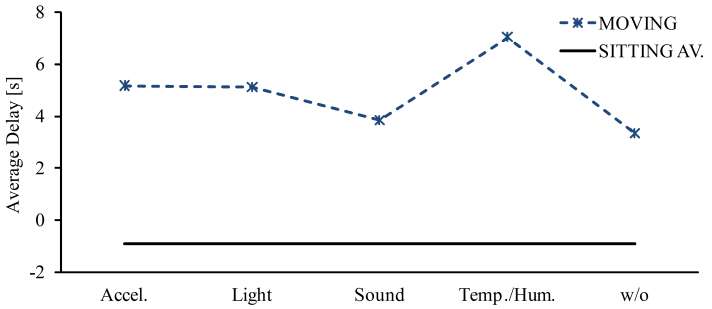

Fig. 8. Average delay in the moving scenarios compared to the mean average delay in the sitting scenario over all seven participants.

Figure 7, detailing the average error rate, and Figure 8, detailing the average delay for the different induced environmental changes, show that environmental disturbances indeed lead

to decreased user performance. The error rates ranging from about 0.19 to 0.45 (av. standard deviation of about 0.24) are higher than the average error rate of about 0.05 (av. standard deviation of about 0.11) in the sitting scenario. Similarly, the delays ranging from 3.34s to 7.02s (av. standard deviation of about 2.91) are higher than the average "delay" of about -0.92s (av. standard deviation of about 4.61) 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 or the high delay variances in the sitting scenario). Thus, with just this sample of seven persons and this first investigation, the results of the field study have to be considered to be preliminary.

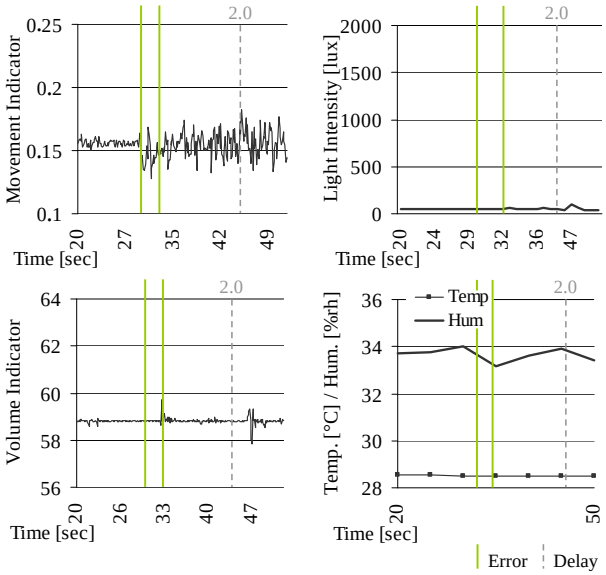

Fig. 9. Example demonstrating how to use sensor values to point at possible reasons for errors and delays (delays given in seconds).

Figure 9 shows an example how the sensor values can be used to reason about the causes of a user's performance. A 30 second extract (seconds 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 activity. Automated usability evaluation tools may now classify these two errors and the delay as possibly caused or at least influenced by movement.

6 Mobile HCI Experiments in the Field

In the second evaluation step, we tested our approach with 19 test persons in the field, that was, the university campus of the University of Vienna. This area allows to change from silent to noisy environments, as well as from shaded areas to sunlight. The test persons followed a guide who determined the movement path. They were instructed by an audio presentation telling the tasks they should perform (see Section 4). The user interactions where logged as well as the environmental conditions which resulted, after filtering of interactions where no sensor values have been recorded, in 940 interactions together with aggregated sensor values for each interaction (see Section 4 for calculation details). The additional, external audio logger generated less values and, therefore, the results for the audio logger are based on a subset of 484 user interactions.

The hardware setup and metrics used are similar to the metrics used in the laboratory study. The scenarios were similar as well, meaning that a first scenario where the test persons were sitting in a shaded area was followed by two movement scenarios. Contrary to the laboratory, no artificial environmental changes were induced, but the second and third scenarios included movement to areas exhibiting different environmental conditions (context settings).

For each user interaction four measures have been introduced for each sensor value to capture average values as well as maximum values, and fluctuations represented by the range of values and the deviation of subsequent values. The following measures are calculated for light, sound and external audio recorder (audio logger), acceleration, temperature, and humidity:

Avg. The average of the sensor readings within the time frame.

Max. The maximum of the sensor readings within the time frame.

Range. The range is here defined as the difference of the maximum and minimum sensor readings within the time frame, i.e., $range = maximum - minimum$.

Dev. The deviation is here defined as the average deviation between two subsequent values.

6.1 Investigation of Errors

The user error rate recorded was 0.32 overall (for audio: 0.36). By sorting the interactions by user input correctness (no error followed by erroneous operations), raw sensor readings, like visualized for for light measures in Figures 10 and 11, were derived. Figure 10 shows the *Avg* values per user interaction ID, while Figure 11 shows the deviation *Dev*, and, thus, the dynamics in light measurements per user interaction ID. On the average, the light values in the error-free cases are higher than the values of the erroneous cases. This result is unexpected, but as derived from the users' questionnaires and interviews, users have quickly reacted to sunlight on their displays and, e.g., tilt the device, which is most likely the reason for this observation.

In order to see whether environmental conditions differ in error-free and erroneous cases, for each environmental condition (i.e., sound, temperature, humidity, acceleration, light, and audio) we calculated the average of each value *val* (i.e., average, maximum, range, and deviation) both for the error-free *Noerror* and the erroneous case *Error*. A first overview of

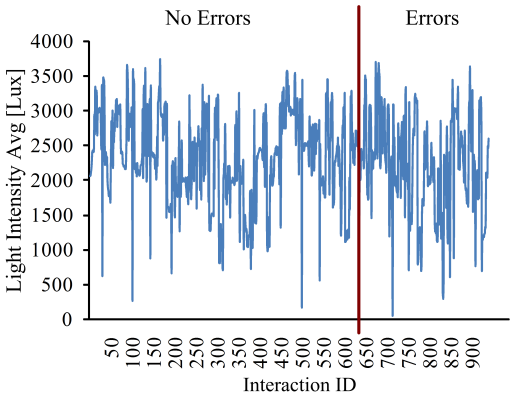

Fig. 10. Light intensity: Average values.

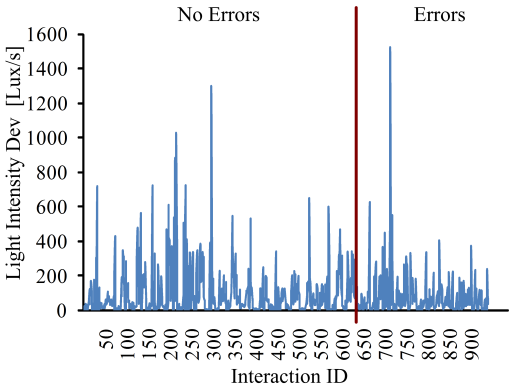

Fig. 11. Light intensity: Deviation values.

the relationship between errors and environmental conditions can be given by calculating the difference of the values as follows:

$$Diff_{val} = Error_{val} - Noerror_{val}.$$

In case $Diff_{val} > 0$, a positive correlation between error and environmental condition is intuitively expected. If $Diff_{val} < 0$, the phenomenon occurred that the test persons were not disturbed, but performed better in environmental stress situations. In case $Diff = 0$, no such influence can be expected. Additionally, we performed a statistical test (TTest) to investigate whether a significant difference between the mean values of the two samples for *Error* and *Noerror* can be stated in each of the 24 cases. In 15 cases, the calculated means differ significantly meaning that the measured environmental conditions were significantly different for the error and error-free cases. For sound *Avg*, humidity *Avg* and *Max*, acceleration nearly all values, light *Range*, and audio *Avg* and *Dev*, a significant difference in means cannot be assumed. This fact is also partly visible in low absolute *Diff* values.

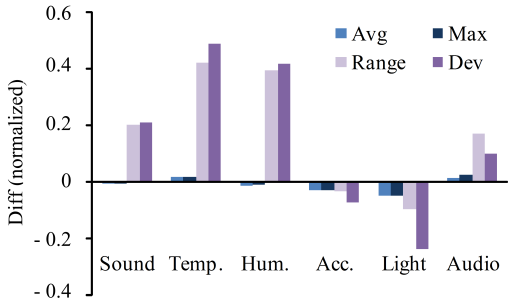

Fig. 12. Difference of Avg, Max, Range, and Dev between the error and the error-free cases.

Figure 12 depicts all difference values (note, that the units were normalized to ratios by dividing the difference values by the maximum value of the error case and error-free case). Table 2 summarizes the difference values derived in original order of magnitude. For

temperature and humidity, we have to say that both values did not change significantly over time. Additionally, these sensors react rather slowly to changes in the environment. Thus, the values measured can not be considered to influence single, particular user interactions (positive *Diff* for all values for temperature, partly negative and positive for humidity). All sound and audio values resulted in positive *Diff* values, which correspond to the verbal feedback of the test participants who reported the distraction by sound. It is remarkable, that not the average values correspond to this feedback but the maximum and range values. Contrary, for light and acceleration values, a negative *Diff* was calculated, meaning that higher movement or light values (e.g., sunlight in this setting) are observed together with good interaction performance (here, we expect a negative statistic correlation). Again, this can be explained by taking the test user feedback into account. Many of the users reported, that they simply shaded the display and moved it out of the sun to be able to work. The negative *Diff* values and small absolute *Diff* values for acceleration/movement reflect the experienced reaction of the users to stop for a very short time to interact with the mobile application.

Table 2. Differences *Diff* between average sensor values in the erroneous and the error-free case.

	Avg	**Max**	**Range**	**Dev**
Sound level	-0.1926	0.0170	0.0317	0.0063
Temp. [degrees C]	0.4786	0.4818	0.0064	0.0061
Hum. [%]	-0.6070	-0.5482	0.1128	0.0928
Acc. [g]	-0.0035	-0.0043	-0.0011	-0.0005
Light [Lux]	-110.69	-126.02	-37.53	-23.47
Audio* [dB]	0.8411	1.6421	1.4347	0.2721

6.2 Investigation of Delays

The 635 error-free user interactions were considered for delays. Since temperature and humidity sensors do not react accurately, we omitted these values for investigating delays. By investigating the correlation between environmental phenomena and delays, first results showed that there is no such simple dependency as with increasing delay, increasing environmental values will be observed. The minimum and maximum delays (negative delays and delays over 15 seconds) can be considered as outliers of the experiment. Most interactions have been solved within a delay of zero up to five seconds.

By relating the delays with the various sensor values we observed different high values for *Dev* and *Range* depending on the delay range. For *Avg* and *Max* we could not find a simple similar pattern. Based on these observations, we propose a classification of delays as given in Table 3 together with the absolute number of occurrences and the relative occurrences and will relate this classes to *Dev* and *Range* values.

Table 3. Delay classes.

	Delay range	**No. of occurrences**	**Rel. occurrences**
Class 1	delay $\leq$ 1 sec.	102	0.16
Class 2	1 sec. $<$ delay $\leq$ 3 sec.	223	0.35
Class 3	3 sec. $<$ delay $\leq$ 5 sec.	182	0.29
Class 4	5 sec. $<$ delay $\leq$ 10 sec.	93	0.15
Class 5	10 sec. $<$ delay	35	0.05

Figures 13 and 14 show the relative average deviation and relative average range values for acceleration, light, sound, and audio (deviation and range normalized, i.e., divided by the

maximum value for deviation, respectively range, in each category). Although these values do not show a simple pattern for all delay classes, in particular for delay class 4, the maximum values have been measured for acceleration, light, sound, and the second highest value for the audio logger. The delays categorized as class 4 delays are likely to be correlated with the observed higher sensor value fluctuations. For reasons of completeness, Table 4 summarizes the original (not normalized) *Dev* and *Range* values together with the CI90 confidence value.

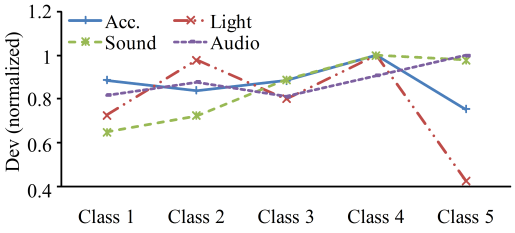

Fig. 13. Deviation related to different delay classes.

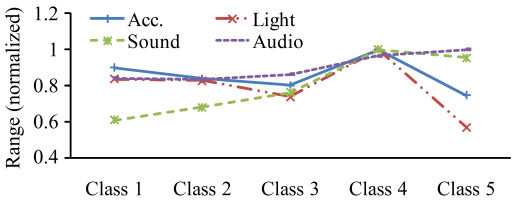

Fig. 14. Range related to different delay classes.

Table 4. *Dev* and *Range* values clustered by classes (+/- CI90 range).

	Acc. [g]	**Light [Lux]**	**Sound [Level]**	**Audio* [dB]**
Dev				
Class 1	0.0064 (+/- 0.00074)	83.63 (+/- 10.86)	0.0188 (+/- 0.0024)	2.36 (+/- 0.4061)
Class 2	0.0061 (+/- 0.00069)	112.42 (+/- 19.09)	0.0209 (+/- 0.0022)	2.53 (+/- 0.3772)
Class 3	0.0064 (+/- 0.00076)	92.11 (+/- 15.30)	0.0256 (+/- 0.0042)	2.35 (+/- 0.3388)
Class 4	0.0073 (+/- 0.00066)	114.97 (+/- 12.94)	0.0288 (+/- 0.0041)	2.62 (+/- 0.3703)
Class 5	0.0055 (+/- 0.00067)	49.36 (+/- 5.00)	0.0282 (+/- 0.0055)	2.89 (+/- 0.4949)
Range				
Class 1	0.0352 (+/- 0.0042)	398.96 (+/- 53.75)	0.1014 (+/- 0.0139)	6.84 (+/- 0.9542)
Class 2	0.0328 (+/- 0.0057)	394.91 (+/- 55.58)	0.1135 (+/- 0.0182)	6.79 (+/- 0.9656)
Class 3	0.0314 (+/- 0.0037)	352.75 (+/- 51.27)	0.1269 (+/- 0.0228)	7.03 (+/- 0.9738)
Class 4	0.0391 (+/- 0.0039)	475.71 (+/- 54.21)	0.1661 (+/- 0.0248)	7.87 (+/- 1.0835)
Class 5	0.0293 (+/- 0.0035)	271.61 (+/- 42.12)	0.1589 (+/- 0.0246)	8.12 (+/- 1.0759)

Since we could not derive simple delay patterns (only for delay class 4 we found a pattern so far), the influences of environmental conditions have to be considered to be less than given for the error rate. It is expected, that user interaction types significantly influence delays due to their complexity and time frame. Hence, we investigated the relation between the different delay classes and the defined four user interaction types (see Section 4). Figure 15 shows the delays for all error-free user interactions sorted by the interaction type. This figure is further a visualization of the number of occurrences of type 1 (69), type 2 (103), type 3 (283), and type 4 (180) error-free user interactions in this set. Figure 16 gives an overview of delay occurrence ratios of the delay classes per user interaction type. For instance, for interaction type 1 (text notes) and 2 (slider manipulations), we observed a higher relative number of occurrences of delay class 4 than for the other types.

From the observation that the different interaction types resulted in different occurrences of delay classes, delays are expected to be not independent from the interaction type. As for delay class 4, interaction types 1 and 2 are potential causes for these delays. On the other hand, due to the high sensor readings in terms of *Dev* and *Range*, we see also an influence of environmental conditions for these high delays. Based on these insights we are looking

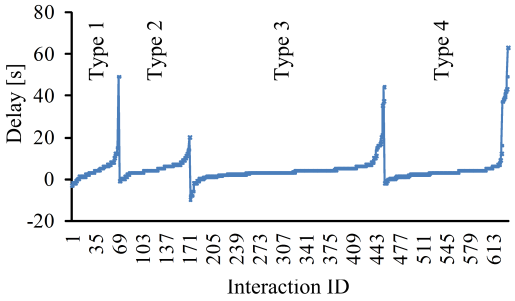

Fig. 15. Delay ordered by interaction type.

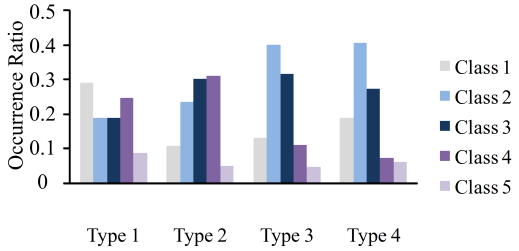

Fig. 16. Delay occurrence ratio per interaction type.

forward to find a sufficient set of additional patterns by relating delays, interaction types, and environmental measures to better reason about possible causes for high delays.

6.3 Qualitative Insights of the Field Usability Evaluation

The qualitative analysis of the usability evaluation was done by analyzing the logfiles generated by MobEval, notes taken by the evaluator during the experiments, videos of the experiments, questionnaires, and structured interviews of the participants. Hereby, both the application and the influence of the context parameters have been addressed. Table 5 shows the different tasks (user interaction types), occurrences, and the occurred errors per interaction type.

Table 5. Occurred errors per task.

Task type	**No. of occurrences**	**Error ratio**
Text note (type 1)	133	0.48
Continuous status event (type 2)	184	0.44
Discrete status event (type 3)	404	0.30
Toggle event (type 4)	219	0.18

Although the tasks were done in different context scenarios, we already could identify that the users had the biggest problems with making a text note and performing the continuous status event (implemented through a slider). This indicates that there is most likely a design problem with these two interaction types resulting in misunderstandings.

The environmental influences temperature and humidity were not in particular realized as being annoying. Concerning light, the users did not report that light (sunlight) was disturbing. Users often simply shaded the display with their body for the time interacting with the mobile device. While moving, it was difficult for the users to enter, for example, text notes during walking. Here, the users simply stopped for a moment (or slowed down movement), which is a cause for the low error rate while moving.

The users stated in interviews that the people around them did not disturb them. The video analysis and the logged data on the other hand indicated that noise and people around them distracted the test users indeed which was reflected in an increased error rate during situations when there was a high noise level and a lot of people were around them. This is especially true during situations when the test person was walking.

Further it could be observed that when people were walking and they had to execute a task

simultaneously they put a lot of focus on the task. Hence, they were not as much distracted by the surrounding environment as expected. This resulted several times in situations that could have caused accidents. For example, the users sometimes almost collided with a street lamp or pillar (if the evaluator would not have prevented this accident).

7 Conclusions and Future Work

To improve usability studies in the field, we proposed not to avoid influencing context changes but to sense them and take them into account during usability evaluation. We 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. The framework was successfully used for collecting sensor data both in a lab and in a field study. Additionally, the mobile application under test logged user interactions which were related to the context sensor values light, sound (and audio logger), temperature, humidity, and acceleration.

We conducted two studies under similar conditions, one in the laboratory and one at a university campus area. In the laboratory study we tested seven users and induced environmental changes, like noise and light, artificially and observed the user reactions. Both error rate and delay of user interactions showed higher values in situations the user was exposed to environmental changes. These encouraging results were partly confirmed by a field experiment conducted with 19 test persons.

Concerning the error rates, higher noise levels were observed for user interactions which resulted in errors. This outcome complies to the qualitative usability evaluation. For light and movement (acceleration), we recognized a reciprocal behavior which we could explain only by taking the qualitative usability evaluation results. Users stated that movement and light were no problems because they could adapt to the conditions (stopping shortly or tilting the smart phone under test in case of sunlight). Temperature and humidity sensors react slowly to environmental changes and cannot be considered to influence single, particular user interactions, which was confirmed by the field evaluation and the qualitative insights, i.e., the users stated that they did not realize temperature or humidity changes. For faster changing environmental conditions, we observed that a condition was perceived as disturbing if the user could not adapt to it (noise), while conditions the users could adapt to, like movement and light, were not experienced as being disturbing.

By investigating the delay of user interactions we experienced that we could not state a simple relation between context parameters and delays. We classified delays and could indeed find context parameter patterns for one particular delay class. Additionally, we observed that the interaction type influences the delays which has to be considered in future evaluations.

The combination of different environmental context factors is still an open issue and future work shall focus on combining, evaluating, and interpreting context factors as multi-variate data, not single, independent values as we have considered so far. The aim is to indicate context situations where a usage error is probable or the risk of delay is high. This information is expected to help usability evaluators to find bad designs issues of a mobile application and, additionally, identify situations where users are highly focused on their task and are exposed to accidents.

References

1. M. Aleksy, T. Butter, and M. Schader. Architecture for the Development of Context-Sensitive Mobile Applications. *Mobile Information Systems*, 4(2):105–117, 2008.
2. M. Assad, D. J. Carmichael, J. Kay, and B. Kummerfeld. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In *Pervasive Computing 2007*, pages 55–72, 2007.
3. 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.
4. 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.
5. H. W. Bristow, C. Baber, J. Cross, J. F. Knight, and S. I. Woolley. Defining and Evaluating Context for Wearable Computing. *International Journal of Human-Computer Studies*, 60(5–6):798–819, 2004.
6. C. Challiol, A. Fortier, S. Gordillo, and G. Rossi. Architectural and Implementation Issues for a Context-Aware Hypermedia Platform. *Mobile Multimedia*, 4(2):118–138, 2008.
7. D. Chen, A. Schmidt, and H.-W. Gellersen. An Architecture for Multi-Sensor Fusion in Mobile Environments. In *International Conference on Information Fusion*, pages 861–868, 1999.
8. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 17–24, 2000.
9. Crossbow Technology Incorporated. *MPR-MIB Users Manual*, June 2006.
10. Crossbow Technology Incorporated. *MTS/MDA Sensor Board Users Manual*, June 2006.
11. H.-L. Duh, G. Tan, and V.-H. Chen. Usability Evaluation for Mobile Device: A Comparison of Laboratory and Field Tests. In *8th Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 181–186, New York, NY, USA, 2006.
12. 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 Conference on Programming Language Design and Implementation*, pages 1–11, 2003.
13. 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.
14. S. Gulliver, G. Ghinea, M. Patel, and T. Serif. A Context-Aware Tour Guide: User Implications. *Mobile Information Systems*, 3(2):71–88, 2007.
15. 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.
16. P. Hu, J. Indulska, and R. Robinson. An Autonomic Context Management System for Pervasive Computing. In *Pervasive 2008*, pages 213–223, 2008.
17. K. A. Hummel, A. Hess, and T. Grill. Environmental Context Sensing for Usability Evaluation in Mobile HCI by Means of Small Wireless Sensor Networks. In *6th International Conference on Advances in Mobile Computing and Multimedia Workshops*, pages 302–306, 2008.
18. J. Kjeldskov and C. Graham. A Review of Mobile HCI Research Methods. In *5th International Symposium on Human-Computer Interaction with Mobile Devices and Services*, pages 317–335, 2003.
19. J. Kjeldskov and M. Skov. Exploring Context-Awareness for Ubiquitous Computing in the Healthcare Domain. *Personal and Ubiquitous Computing*, 11(7):549–562, 2007.
20. 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.
21. S. Pedell, C. Graham, J. Kjeldskov, and J. Davies. Mobile Evaluation: What the Data and the Metadata Told Us. In *OZCHI 2003*, pages 96–105, 2003.
22. K. Pousttchi and B. Thurnher. Understanding Effects and Determinants of Mobile Support Tools: A Usability-Centered Field Study on IT Service Technicians. In *ICMB '06: International Confer-*

ence on Mobile Business, page 10, 2006.
23. C. Ryan and A. Gonsalves. The Effect of Context and Application Type on Mobile Usability: An Empirical Study. In *ACSC '05: 28th Australasian Conference on Computer Science*, pages 115–124, 2005.
24. D. Salber, A. Dey, and G. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 434–441, 1999.