Chapter XIII
Context-Awareness and Mobile Devices

Anind K. Dey
Carnegie Mellon University, USA

Jonna Häkkilä
Nokia Research Center, Finland

ABSTRACT

Context-awareness is a maturing area within the field of ubiquitous computing. It is particularly relevant to the growing sub-field of mobile computing as a user's context changes more rapidly when a user is mobile, and interacts with more devices and people in a greater number of locations. In this chapter, we present a definition of context and context-awareness and describe its importance to human-computer interaction and mobile computing. We describe some of the difficulties in building context-aware applications and the solutions that have arisen to address these. Despite these solutions, users have difficulties in using and adopting mobile context-aware applications. We discuss these difficulties and present a set of eight design guidelines that can aid application designers in producing more usable and useful mobile context-aware applications.

INTRODUCTION

Over the past decade, there has been a widespread adoption of mobile phones and personal digital assistants (PDAs) all over the world. Economies of scale both for the devices and the supporting infrastructure have enabled billions of mobile devices to become affordable and accessible to large groups of users. Mobile computing is a fully realized phenomenon of everyday life and is the first computing platform that is truly ubiquitous. Technical enhancements in mobile computing, such as component miniaturization, enhanced computing power, and improvements in supporting infrastructure have enabled the creation of more versatile, powerful, and sophisticated mobile devices. Both industrial organizations and academic researchers, recognizing the powerful combina-

tion of a vast user population and a sophisticated computing platform, have focused tremendous effort on improving and enhancing the experience of using a mobile device.

Since its introducion in the mid-1980s, the sophistication of mobile devices in terms of the numbers and types of services they can provide has increased many times over. However, at the same time, the support for accepting input from users and presenting output to users has remained relatively impoverished. This has resulted in slow interaction, with elongated navigation paths and key press sequences to input information. The use of predictive typing allowed for more fluid interaction, but mobile devices were still limited to using information provided by the user and the device's service provider. Over the past few years, improvements to mobile devices and back-end infrastructure has allowed for additional information to be used as input to mobile devices and services. In particular, context, or information about the user, the user's environment and the device's context of use, can be leveraged to expand the level of input to mobile devices and support more efficient interaction with a mobile device. More and more, researchers are looking to make devices and services *context-aware*, or adaptable in response to a user's changing context.

In this chapter, we will define context-awareness and describe its importance to human-computer interaction and mobile devices. We will describe some of the difficulties that researchers have had in building context-aware applications and solutions that have arisen to address these. We will also discuss some of the difficulties users have in using context-aware applications and will present a set of design guidelines that indicate how mobile context-aware applications can be designed to address or avoid these difficulties.

What is Context-Awareness

The concept of context-aware computing was introduced in Mark Weiser's seminal paper 'The Computer for the 21st Century' (Weiser, 1991). He describes ubiquitous computing as a phenomenon '*that takes into account the natural human environment and allows the computers themselves to vanish into the background.*' He also shapes the fundamental concepts of context-aware computing, with computers that are able to capture and retrieve context-based information and offer seamless interaction to support the user's current tasks, and with each computer being able to *'adapt its behavior in significant ways'* to the captured context.

Schilit and Theimer (1994a) first introduce the term *context-aware computing* in 1994 and define it as software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time." We prefer a more general definition of context and context-awareness:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves, and by extension, the environment the user and applications are embedded in. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. (Dey, 2001)

Context-aware features include using context to:

- Present information and services to a user
- Automatically execute a service for a user and
- Tag information to support later retrieval

In supporting these features, context-aware applications can utilize numerous different kinds of information sources. Often, this information comes from sensors, whether they are software sensors detecting information about the networked, or virtual, world, or hardware sensors detecting information about the physical world. Sensor data can be used to recognize the usage situation for instance from illumination, temperature, noise

level, and device movements (Gellersen, Schmidt & Beigl, 2002; Mäntyjärvi & Seppänen, 2002). Typically, sensors are attached to a device and an application on the device locally performs the data analysis, context-recognition, and context-aware service.

Location is the most commonly used piece of context information, and several different location detection techniques have been utilized in context-awareness research. Global positioning system (GPS) is a commonly used technology when outdoors, utilized, for example, in car navigation systems. Network cellular ID can be used to determine location with mobile phones. Measuring the relative signal strengths of Bluetooth and WLAN hotspots and using the hotspots as beacons are frequently used techniques for outdoors and indoors positioning (Aalto, Göthlin,Korhonen et al., 2004; Burrell & Gay, 2002; Persson et al., 2003). Other methods used indoors include ultrasonic or infrared-based location detection (Abowd et al., 1997; Borriello et al., 2005).

Other commonly used forms of context are time of day, day of week, identity of the user, proximity to other devices and people, and actions of the user (Dey, Salber & Abowd, 2001; Osbakk & Rydgren, 2005). Context-aware device behavior may not rely purely on the physical environment. While sensors have been used to directly provide this physical context information, sensor data often needs to be interpreted to aid in the understanding of the user's goals. Information about a user's goals, preferences, and social context can be used for determining context-aware device behavior as well. Knowledge about a user's goals helps prioritize the device actions and select the most relevant information sources. A user's personal preferences can offer useful information for profiling or personalizing services or refining information retrieval. The user may also have preferences about quality of service issues such as cost-efficiency, data connection speed, and reliability, which relate closely to mobile connectivity issues dealing with handovers and alternative data transfer mediums. Finally, social context forms an important type of context as mobile devices are commonly used to support communication between two people and used in the presence of other people.

Relevance to HCI

When people speak and interact with each other, they naturally leverage their knowledge about the context around them to improve and streamline the interaction. But, when people interact with computers, the computing devices are usually quite ignorant of the user's context of use. As the use of context essentially expands the conversational bandwidth between the user and her application, context is extremely relevant to human-computer interaction (HCI). Context is useful for making interaction more efficient by not forcing users to explicitly enter information about their context. It is useful for improving interactions as context-aware applications and devices can offer more customized and more appropriate services than those that do not use context. While there have been no studies of context-aware applications to validate that they have this ability, anecdotally, it is clear that having more information about users, their environments, what they have done and what they want to do, is valuable to applications. This is true in network file systems that cache most recently used files to speed up later retrieval of those files, as well as in tour guides that provide additional information about a place of interest the user is next to.

Relevance to Mobile HCI

Context is particularly relevant in mobile computing. When users are mobile, their context of use changes much more rapidly than when they are stationary and tied to a desktop computing platform. For example, as people move, their location changes, the devices and people they interact with changes more frequently, and their goals and needs change. Mobility provides additional opportunities for leveraging context but also requires additional context to try and understand how the user's goals are changing. This places extra burden on the mobile computing platform, as it needs to sense potentially rapidly changing context, synthesize it and act upon it. In the next section, we will discuss the difficulties that application builders have had with building context-aware applica-

tions and solutions that have arisen to address these difficulties.

BUILDING MOBILE CONTEXT-AWARE APPLICATIONS

The first context-aware applications were centered on mobility. The Active Badge location system used infrared-based badges and sensors to determine the location of workers in an indoor location (Want et al., 1992). A receptionist could use this information to route a phone call to the location of the person being called, rather than forwarding the phone call to an empty office. Similarly, individuals could locate others to arrange impromptu meetings. Schilit, Adams and Want,(1994b) also use an infrared-based cellular network to location people and devices, the PARCTAB, and describe 4 different types of applications built with it (Schilit et al., 1994b). This includes:

- **Proximate selection:** Nearby objects like printers are emphasized to be easier to select than other similar objects that are further away from the user;
- **Contextual information and commands:** Information presented to a user or commands parameterized and executed for a user depend on the user's context;
- **Automatic contextual reconfiguration:** Software is automatically reconfigured to support a user's context; and
- **Context-triggered actions:** If-then rules are used to specify what actions to take based on a user's context.

Since these initial context-aware applications, a number of common mobile context-aware applications have been built: tour guides (Abowd et al., 1997; Cheverst et al. 2000; Cheverst, Mitchell & Davies, 2001), reminder systems (Dey & Abowd, 2000; Lamming & Flynn, 1994) and environmental controllers (Elrod et al., 1993; Mozer et al., 1995). Despite the number of people building (and re-building) these applications, the design and implementation of a new context-aware application required significant effort, as there was no reusable support for building context-aware applications. In particular, the problems that developers faced are:

- Context often comes from non-traditional devices that developers have little experience with, unlike the mouse and keyboard.
- Raw sensor data is often not directly useful to an application, so the data must be abstracted to turn it into useful context.
- Context comes from multiple distributed and heterogeneous sources, and this context often needs to be combined (or fused) to be useful. This process often results in uncertainty that needs to be handled by the application.
- Context is, by its very nature, dynamic, and changes to it must be detected in real time and applications must adjust to these constant changes in order to provide a positive user experience to users.

These problems resulted in developers building every new application from scratch, with little reuse of code or design ideas between applications.

Over the past five years or so, there has been a large number of research projects aimed at addressing these issues, most often trying to produce a reusable toolkit or infrastructure that makes the design of context-aware applications easier and more efficient. Our work, the Context Toolkit, used a number of abstractions to ease the building of applications. One abstraction, the context widget is similar to a graphical user interface widget in that it abstracts the source of an input and only deals with the information the source produces. For example, a location widget could receive input from someone manually entering information, a GPS device, or an infrared positioning system, but an application using a location widget does not have to deal with the details of the underlying sensing technology, only with the information the sensor produces: identity of the object being located, its location and the time when the object was located. Context interpreters support the interpretation, inference and fusion of context. Context aggregators collect all context-related to

a specific location, object or person for easy access. With these three abstractions, along with a discovery system to locate and use the abstractions, an application developer no longer needs to deal with common difficulties in acquiring context and making it useful for an application, and instead can focus on how the particular application she is building can leverage the available context. Other similar architectures include JCAF (Bardram, 2005), SOCAM (Gu, Pung & Zhang, 2004), and CoBRA (Chen et al., 2004).

While these architectures make mobile context-aware applications easier to build, they do not address all problems. Outstanding problems needing support in generalized toolkits include representing and querying context using a common ontology, algorithms for fusing heterogeneous context together, dealing with uncertainty, and inference techniques for deriving higher level forms of context such as human intent. Despite these issues, these toolkits have supported and continue to support the development of a great number of context-aware applications. So, now that we can more easily build context-aware applications, we still need to address how to design and build *usable* mobile context-aware applications. We discuss this issue in the following section.

USABILITY OF MOBILE CONTEXT-AWARE APPLICATIONS

With context information being provided as implicit input to applications and with those applications using this context to infer human intent, there are greater usability concerns than with standard applications that are not context-aware. Bellotti and Edwards discuss the need for context-aware applications to be *intelligible*, where the inferences made and actions being taken are made available to end-users (Bellotti & Edwards, 2001). Without this intelligibility, users of context-aware applications would not be able to decide what actions or responses to take themselves (Dourish, 1997).

To ground our understanding of these abstract concerns, we studied the usability and usefulness of a variety of context-aware applications (Barkhuus & Dey, 2003a; 2003b). We described a number of real and hypothetical context-aware applications and asked subjects to provide daily reports on how they would have used each application each day, whether they thought the applications would be useful, and what reservations they had about using each application. All users were given the same set of applications, but users were split into three groups with each group being given applications with a different level of proactivity. One group was given applications that they would personalize to determine what the application should do for them. Another group was provided with information about how their context was changing, and the users themselves decided how to change the application behavior. The final group was evaluating applications that autonomously changed their behavior based on changing context. Additional information was also gathered from exit interviews conducted with subjects.

Users indicated that they would use and prefer applications that had higher degrees of proactivity. However, as the level of proactivity increased, users had increasing feelings that they were losing control. While these findings might seem contradictory, it should be considered that owning a mobile phone constitutes some lack of control as the user can be contacted anywhere and at anytime; the user may have less control but is willing to bear this cost in exchange for a more interactive and smoother everyday experience. Beyond this issue of control, users had other concerns with regards to the usability of context-aware applications. They were concerned by the lack of feedback, or intelligibility, that the applications provided. Particularly for the more proactive versions of applications, users were unclear how they would know that the application was performing some action for them, what action was being performed, and why this action was being performed. A third concern was privacy. Users were quite concerned that the context data that was being used on mobile platforms could be used by service providers and other entities to track their location and behaviors. A final concern that users had was related to them evaluating

multiple context-aware applications. With potentially multiple applications vying for a user's attention, users had concerns about information overload. Particularly when mobile and focusing on some other task, it could be quite annoying to have multiple applications on the mobile device interrupting and requesting the user's attention simultaneously or even serially.

In the remainder of this chapter, we will discuss issues for designing context-aware applications that address usability concerns such as these.

Support for Interaction Design

Despite all of the active research in the field of context-aware computing, much work needs to be done to make context-awareness applications an integral part of everyday life. As context-awareness is still a very young field, it does not have established design practices that take into account its special characteristics. The development of applications has so far been done primarily in research groups that focus more on proof-of-concept and short-term use rather than deployable, long-term systems. For most of these applications, the interaction design has rarely been refined to a level that is required for usable and deployable applications. Particularly for applications aimed at consumers and the marketplace, robustness, reliability and usability must be treated more critically than they are currently, as these factors will have a significant impact on their success.

Currently, the lack of existing high-quality, commercial, and publicly available applications limits our ability to assess and refine the best practices in interaction design of context-aware mobile applications. As there is very little experience with real-life use of these applications, the ability of developers to compare and iterate on different design solutions is very restricted. As user groups for a particular application mostly do not exist yet, much of the current research is based on hypothesized or simulated systems rather that actualized use situations. Knowledge of what device features people fancy and which they just tolerate, and when application features become insignificant or annoying, are issues that are hard to anticipate without studies of long-term real-life usage.

As with any other novel technology, bringing it to the marketplace will bring new challenges. Bringing context-awareness to mobile devices as an additional feature may lead to situations where the interaction design is performed by people with little experience in context-aware computing. Using well-established commercial platforms such as mobile phones or PDAs often means that user interface designers only have experience with conventional mobile user interfaces. On the other hand, the technical specifications of an application are often provided by people who have no expertise in human-computer interaction issues. When entering a field that involves interdisciplinary elements, such as mobile context-awareness, providing tools and appropriate background information for designers helps them to recognize the risks and special requirements of the technology.

Hence, there are several factors which make examining context-awareness from the usability and interaction design perspective relevant. Failures in these may lead not only to unprofitable products, but may result in an overall negative effect—they may slow down or prevent the underlying technology from penetrating into mass markets.

Usability Risks for Mobile Context-Aware Applications

A system and its functionality are often described with mental models that people form from using the system. According to Norman (1990), one can distinguish between the designer's mental model and the user's mental model. The designer's model represents the designer's understanding and idea of the artefact being constructed, whereas the user's model is the user's conceptual model of the same artefact, its features and functionality, which has developed through her interaction with the system. In order to respond to the user's needs, efficiently fulfil the user's goals and satisfy the user's expectations, the designer's and user's understanding of the device or application should be consistent with each other, in other words, the user's model and designer's model should be the same (Norman, 1990).

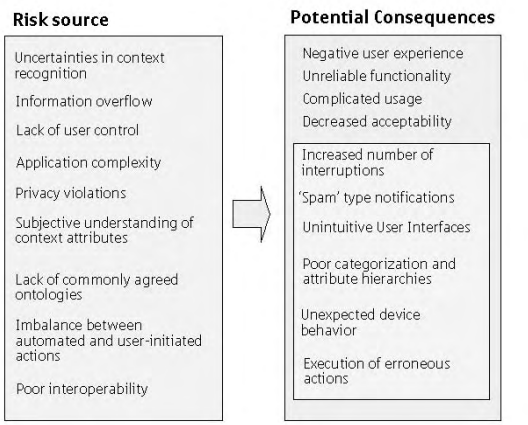
To ensure the best possible result, the mental models of different stakeholders in application development and use have to meet each other. First, the mental models of the application's technical designer and user interface designer should be consistent. This means that the user interface designer should have a basic understanding of the special characteristics of context-aware technology. Second, the designer's and user's mental models of the application should be the same. People's perception of context may differ significantly from each other, and both attributes and the measures used to describe context may vary greatly (Hiltunen, Häkkilä & Tuomela, 2005; Mäntyjärvi et al., 2003). The relationship between the designer's and the user's mental models should be checked with user tests several times during the design process. Without this careful design, there are two significant usability risks that may result: users will be unable to explain the behavior of the context-aware application, nor predict how the system will respond given some user action. While this is true of all interactive systems, it is especially important to consider for context-aware systems as the input to such systems is often implicit.

Context-awareness has several characteristics that can be problematic in interaction design. Figure 1 summarizes potential usability risks with context-aware applications.

A fundamental cause of potential usability risks is *uncertainty in context recognition*, which can be due to different reasons, such as detection accuracy, information fusion, or inferring logic. This is a key issue for designing the user interface for a mobile context-aware application, as it affects the selected features, their functionality and accuracy. In practice, features such as the proactivity level may be designed differently if the confidence level in context recognition can be estimated correctly. Uncertainty is a part of the nature of context-aware applications. Thus, it is important that the application and UI designers share a common understanding of the matter and take it into account when designing both the application and its user interface.

Application complexity has a tendency to grow when functions are added and it forms a potential risk for context-aware applications, as they use a greater number of information sources than traditional mobile applications. Hiding the complex nature of the technology while maintaining a sufficient level of feedback and transparency so that the user can still make sense of the actions the device is performing (i.e., intelligibility) is a challenging issue. Here, the involvement of user-

Figure 1. Sources of usability risks and their potential consequences related to context-aware mobile applications. Consequences that are unique to context-aware mobile applications are in the smaller rectangle on the right.



centric design principles is emphasized. Usability testing and user studies performed in an authentic environment combined with iterative design are key elements to producing well-performing user interface solutions.

Poor interoperability of services and applications relates to the absence of standardization in this maturing field and it limits the application design, available services, and seamless interaction desired across a wide selection of devices and users. Interoperability issues have gained much attention with the current trend of mobile convergence, where different mobile devices resemble each other more and more, yet providing services for them must be performed on a case-by-case basis.

Subjective understanding of context attributes creates a problem for user interface design, as the measures, such as the light intensity or noise level in everyday life are not commonly understood by end-users in terms of luxes or decibels but in relative terms such as 'dark,' 'bright,' 'silent' or 'loud.' This issue is connected to the *lack of commonly agreed ontologies*, which would guide the development of context-aware applications. The difficulties in categorizing context attributes and modeling context is evident from the literature (Hiltunen, Häkkilä & Tuomela, 2005; Mäntyjärvi et al., 2003).

As indicated earlier, *privacy violations* are possible with mobile context-aware systems collecting, sharing and using a tremendous amount of personal information about a user. When such information is shared with a number of different services, each of which will be contacting the user, *information overflow* often results. One can imagine a potential flow of incoming advertisements when entering a busy shopping street, if every shop within a radius of one hundred meters was to send an advertisement to the device. Information overflow is particularly a problem for the small screens that are typical with handheld devices.

As our earlier studies illustrated, the *lack of user control* can easily occur with mobile device automation, when context-triggered actions are executed proactively. However, the promise of context-awareness is that it provides "ease of use" by taking over actions that the user does not want to do or did not think to do for themselves. Any solution for correcting the *imbalance between the set of automated actions and user-initiated actions*, must take user control into account.

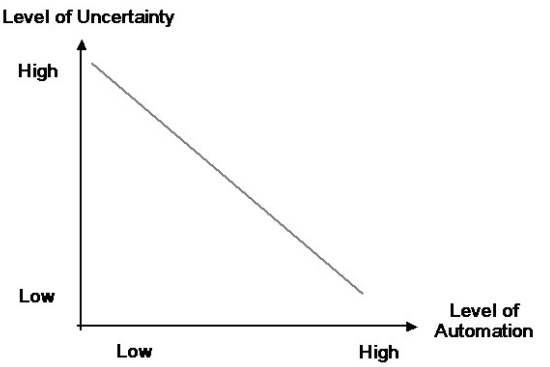
The consequences resulting from these usability risks are numerous. The general outcome can be a negative user experience. This may result from an increased number of interruptions, spam, and the execution of erroneous or otherwise unintuitive device behavior. Unreliable device functionality, and unintelligible user interfaces can lead to reduced acceptability of context-aware applications in the marketplace.

Design Guidelines for Mobile Context-Aware Applications

Context-awareness typically contains more risks than conventional, non-context-aware technology. At the same time, context-awareness can offer much added value to the user. In order to provide this value to end-users and avoid these negative design consequences and minimize usability risks, we have sought to provide a set of design guidelines that can offer practical help for designers who are involved in developing context-aware mobile applications (Häkkilä & Mäntyjärvi, 2006). These general guidelines have been validated in a series of user studies (Häkkilä & Mäntyjärvi, 2006) and should be taken into account when selecting the features of the application and during the overall design process.

GL1. Select appropriate level of automation. A fundamental factor with context-awareness is that it incorporates uncertainty. Uncertainty in context-recognition is caused by several different sources, such as detection accuracy, information fusion, or inferring logic. This is a key issue in designing user interfaces, as it affects the selected features, their functionality and accuracy. In practice, features such as the automation level or level of proactivity may be designed differently if the confidence level of context recognition can be estimated correctly. The relationship between uncertainty and selected application automation level is illustrated in Figure 2. As shown in Fig-

Figure 2. How uncertainty in context-recognition should affect the selected level of automation/proactivity



ure 1, *uncertainties in context recognition* create significant usability risks, however, by selecting an appropriate level of automation, an application designer can acknowledge this fact and address it appropriately. The greater the uncertainty is in the context-recognition, the more important it is not to automate actions. The automation level has also a direct relationship with user control, and its selection has a large impact on the number of expected interruptions the system creates for the user. The level of automation must be considered in relation to the overall application design, as it affects numerous issues in the user interface design.

GL 2. Ensure user control. The user has to maintain the feeling that he is in the control over the device. The user, who normally has full control over his mobile device, has voluntarily given some of it back to the device in order to increase the ease of use of the device. To address this *lack of user control*, an important usability risk, the user must be able to take control of the device and context-aware application at any time. The desire to take control can happen in two basic circumstances—either the device is performing erroneous actions and the user wants to take a correcting action, or the user just wishes to feel in control (a feeling that users often have). The user has to have enough knowledge of the context-aware application and the device functionality in order to recognize malfunctioning behavior, at least in the case where context-recognition errors lead to critical and potentially unexpected actions. The perception of user control is diminished if the device behaves in unexpected manner or if the user has a feeling that the device is performing actions without him knowing it. User control can be implemented, for example, with confirmation dialogues however, this must be balanced with the need to minimize unnecessary interruptions, our next guideline.

GL3. Avoid unnecessary interruptions. Every time the user is interrupted, she is distracted from the currently active task, impacting her performance and satisfaction with the system. In most cases, the interruption leads to negative consequences, however if the system thinks that the interruption will provide high value or benefit to the user, allowing the interruption is often seen as positive. Examples of this are reminders and alarm clocks. The user's interruptibility depends on her context and the user's threshold for putting up with intrusion varies with each individual and her situation. Some context-aware functionality is so important that the user may want the application to override all other ongoing tasks. This leads to a tension between avoiding unnecessary interruptions and supporting user control (GL2).

GL4. Avoid information overflow. The throughput of the information channel to each user is limited, and users can fully focus only on a small number of tasks at one time. In order to address

the usability risk of *information overflow* where several different tasks or events compete for this channel, a priority ordering needs to be defined. Also, the threshold for determining the incoming event's relevancy in the context must be considered in order to avoid unnecessary interruptions (GL3). Systems should not present too much information at once, and should implement filtering techniques for to avoid messages that may appear to be spam to users. Also, information should be arranged in a meaningful manner to maintain and maximize the understandability of the system.

GL 5. Appropriate visibility level of system status. The visibility level of what the system is doing has to be sufficient for the user to be aware of the application's actions. While this guideline has been co-opted from Nielsen and Molich's user interface heuristics (1990), it has special meaning in context-aware computing. The implicit nature of context-awareness and natural *complexity of these types of applications* means that users may not be aware of changes in context, system reasoning or system action. When uncertainty in context-awareness is involved, there must be greater visibility of system state in order to allow the user to recognize the risk level and possible malfunctions. Important actions or changes in context should also be made visible and easily understandable for the user, despite the fact that users may have *subjective understandings of context attributes* and that there may be *no established ontology*. System status need not be overwhelming and interrupting to the user but can be provided in an ambient or peripheral fashion, where information is dynamically made more visible as the importance value grows, and may eventually lead to an interruption event to the user if its value is high enough.

GL 6. Personalization for individual needs. Context-awareness should allow a device or application to respond better to the individual user's personal needs. For instance, an application can implement filtering of interruptions according to the user's personal preferences. Personalization may also be used to improve the subjective understanding of context attributes. Allowing the user to name or change context attributes, such as location names or temperature limits, may contribute to better user satisfaction and ease of use. User preferences may change over time, and their representation in the application can be adjusted, for example implicitly with learning techniques or explicitly with user input settings.

GL 7. Secure user's privacy. *Privacy* is a central theme with personal devices, especially with devices focused on supporting personal communication, and impacts, for example trust, frequency of use, and application acceptability. Special care should be taken with applications that employ context sharing. Privacy requirements often vary between who is requesting the information, the perceived value of the information being requested and what information is being requested, so different levels of privacy should be supported. If necessary, users should have the ability to easily specify that they wish to remain anonymous with no context shared with other entities.

GL 8. Take into account the impact of social context. The social impact of a context-aware application taking an action must be part of the consideration in deciding whether to take the action or not. The application and its behavior reflects on users themselves. In some social contexts, certain device or user behavior may be considered awkward or even unacceptable. In such situations, there must be an appropriate *balance of user-initiated and system-initiated actions.* Social context has also has an effect on interruptibility. For example, an audible alert may be considered as inappropriate device behavior in some social contexts.

Once an application has been designed with these guidelines, the application must still be evaluated to ensure that the usability risks that have been identified for mobile context-aware systems have been addressed. This evaluation can take place in the lab, but is much more useful when conducted under real, *in situ*, conditions.

SUMMARY

Context-aware mobile applications, applications that can detect their users' situations and adapt their behavior in appropriate ways, are an important new form of mobile computing. Context-aware-

ness has been used to overcome the deficit of the traditional problems of small screen sizes and limited input functionalities of mobile devices, to offer shortcuts to situationally-relevant device functions, and to provide location sensitive device actions and personalized mobile services.

Context-awareness as a research field has grown rapidly during recent years, concentrating on topics such as context-recognition, location-awareness, and novel application concepts. Several toolkits for enabling building context-aware research systems have been introduced. Despite their existence, there exist very few commercial or publicly available applications utilizing context-awareness. However, the multitude of research activities in mobile context-awareness allow us to make reasonable assumptions about tomorrow's potential applications. For example, navigation aids, tour guides, location-sensitive and context-sensitive notifications and reminders, automated annotation and sharing of photographs, use of metadata for file annotation, sharing or search are topics which frequently appear in the research literature and will likely be relevant in the future. In addition, using context-awareness to address the needs of special user groups, for example in the area of healthcare also appears to be a rich area to explore.

Despite the active research in context-awareness, there is much that remains to be addressed in interaction design and usability issues for context-aware mobile applications. Due the novelty of the field and lack of existing commercial applications, design practices for producing usable and useful user interfaces have not yet evolved, and end-users' experiences with the technology are not always positive. We have presented a set of 8 design guidelines which have been validated and evaluated in a series of user studies, which point to areas where user interface designers must focus efforts in order to address the usability issues that are commonly found with mobile context-aware applications.

While context-aware applications certainly have more usability risks than traditional mobile applications, the potential benefits they offer to end-users are great. It is important that application designers and user interface designers understand each other's perspectives and the unique opportunities and pitfalls that context-aware systems have to offer. With context-aware applications, careful application and interface design must be emphasized. The consequences resulting from usability risks include an overall negative user experience. Unsuccessful application design may result in diminished user control, increased number of interruptions, spam, and the execution of erroneous device actions or otherwise unintuitive behaviour. Unreliable device functionality and an unintuitive user interface can lead to decreased acceptability of the context-aware features in the marketplace.

In this chapter we have discussed the notion of context-awareness and its relevance to both mobile computing and interaction design in mobile computing. We have described technical issues involved in building context-aware applications and the toolkits that have been built to address these issues. Despite the existence of these toolkits in making context-aware applications easier to build, there are several additional issues that must be addressed in order to make mobile context-aware applications usable and acceptable to end-users. We have presented a number of design guidelines that can aid the designers of mobile context-aware applications in producing applications with both novel and useful functionality for these end-users.

REFERENCES

Aalto, L, Göthlin, N., Korhonen, J., & Ojala T. (2004). Bluetooth and WAP Push based location-aware mobile advertising system. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services* (pp. 49-58).

Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R.. & Pinkerton, M. (1997). Cyberguide: a mobile context-aware tour guide. *ACM Wireless Networks, 3*, 421-433.

Bardram, J. (2005). The java context awareness framework (JCAF)—A service infrastructure

and programming framework for context-aware applications. In *Proceedings of Pervasive 2005* (pp. 98-115).

Barkhuus, L., & Dey, A.K. (2003a). Is context-aware computing taking control away from the user? Three levels of interactivity examined. In *Proceedings of UBICOMP 2003* (pp. 149-156).

Barkhuus, L., & Dey, A.K. (2003b). Location-based services for mobile telephony: A study of users' privacy concerns. In *Proceedings of INTERACT 2003* (pp. 709-712).

Bellotti, V., & Edwards, K. (2001). Intelligibility and accountability: Human considerations in context-aware systems. *HCI Journal, 16*, 193-212.

Borriello, G., Liu, A., Offer, T., Palistrant, C., & Sharp, R. (2005). WALRUS: Wireless, Acoustic, Location with Room-Level Resolution using Ultrasound. In *Proceedings of the 3rd International Conference on Mobile systems, application and services (MobiSys'05),* (pp. 191-203).

Burrell, J., & Gay, G. K. (2002). E-graffiti: Evaluating real-world use of a context-aware system. *Interacting with Computers, 14*, 301-312.

Chen, H., Finin, T. and Joshi, A. Chen, H., Finin, T., & Joshi, A. (2004). Semantic Web in the Context Broker Architecture. (2004). In *Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications (Percom'04),* (pp. 277-286).

Cheverst, K., Davies, N., Mitchell, K., & Friday, A. (2000). Experiences of developing and deploying a context-aware tourist guide: The GUIDE project. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom),* (pp. 20-31).

Cheverst, K., Mitchell, K., & Davies, N. (2001). Investigating Context-Aware Information Push vs. Information Pull to Tourists. In *Proceedings of MobileHCI'01,*

Dey, A.K., & Abowd, G.D. (2000). CybreMinder: A context-aware system for supporting reminders. In *Proceedings of the International Symposium on Handheld and Ubiquitous Computing* (pp. 172-186).

Dey, A.K., Salber, D., & Abowd, G.D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal 16*(2-4), (pp. 97-166).

Dourish, P. (1997). Accounting for system behaviour: Representation, reflection and resourceful action. In Kyng and Mathiassen (Eds.), *Computers and design in context* (pp. 145-170). Cambridge, MA: MIT Press.

Elrod, S., Hall, G., Costanza, R., Dixon, M., & des Rivieres, J. Responsive office environments. *Communications of the ACM* 36(7), 84-85.

Gellersen, H.W., Schmidt, A., & Beigl, M. (2002). Multi-sensor sontext-awareness in mobile devices and smart artefacts. *Mobile Networks and Applications, 7,* 341-351.

Gu, T., Pung, H.K., & Zhang, D.Q. (2004). A middleware for building context-aware mobile services. In *Proceedings of IEEE Vehicular Technology Conference* (pp. 2656-2660).

Häkkilä, J., & Mäntyjärvi, J. (2006). Developing design guidelines for context-aware mobile applications. In *Proceedings of the IEE International Conference on Mobile Technology, Applications and Systems.*

Hiltunen, K.-M., Häkkilä, J., & Tuomela, U. (2005). Subjective understanding of context attributes – a case study. In *Proceedings of Australasian Conference of Computer Human Interaction (OZCHI) 2005,* (pp. 1-4).

Lamming, M., & Flynn. M. (1994). Forget-me-note: Intimate computing in support of human memory. In *Proceedings of Friend21: International Symposium on Next Generation Human Interface* (pp. 125-128).

Mäntyjärvi, J., & Seppänen, T. (2002). Adapting applications in mobile terminals using fuzzy context information. In *Proceedings of Mobile HCI 2002* (pp. 95-107).

Mäntyjärvi, J., Tuomela, U., Känsälä, I., & Häkkilä, J. (2003). Context Studio—Tool for Personalizing Context-Aware Application in Mobile Terminals. In *Proceedings of Australasian Conference of Computer Human Interaction (OZCHI) 2003* (pp. 64-73).

Mozer, M.C., Dodier, R.H., Anderson, M., Vidmar, L., Cruickshank III, R.F., & Miller, D. The Neural Network House: An Overview. In L. Niklasson & M. Boden (Eds.), *Current trends in connectionism*, (pp. 371-380). Hillsdale, NJ: Erlbaum.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of CHI 1990* (pp. 249-256).

Norman, D. A. (1990). *The design of everyday things.* New York, NY: Doubleday.

Osbakk, P., & Rydgren, E. (2005). Ubiquitous computing for the public. In *Proceedings of Pervasive 2005 Workshop on Pervasive Mobile Interaction Devices (PERMID 2005),* (pp. 56-59).

Persson, P., Espinoza, F., Fagerberg, P., Sandin, A., & Cöster, R. (2003). GeoNotes: A Location-based information System for Public Spaces. In K. Hook, D. Benyon & A. Munro (Eds.), *Readings in Social Navigation of Information Space* (pp. 151-173). London, UK: Springer-Verlag.

Schilit, B., & Theimer, M. (1994a). Disseminating active map information to mobile hosts. *IEEE Computer 8*(5), 22-32.

Schilit, B., Adams, N., & Want, R. (1994b). Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (pp. 85-90).

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The Active Badge Location System. *ACM Transactions on Information Systems, 10*(1), 91-102.

Weiser, M. (1991). The computer for 21st century. *Scientific American,* 265(3), 94-104.

KEY TERMS

Context: Any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves, and by extension, the environment the user and applications are embedded in.

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

Design Guidelines: Guidelines or principles that, when followed, can improve the design and usability of a system.

Interaction Design: The design of the user interface and other mechanism that support the user's interaction with a system, including providing input and receiving output.

Mobile Context-Awareness: Context-awareness for systems or situations where the user and her devices are mobile. Mobility is particularly relevant for context-awareness as the user's context changes more rapidly when mobile.

Usability Risks: Risks that result from the use of a particular technology (in this case, context-awareness) that impact the usability of a system.