Enabling the Use of Context in Interactive Applications

Anind K. Dey
College of Computing & Graphics, Visualization, & Usability Center
Georgia Institute of Technology
Atlanta, GA 30332-0280
+1-404-894-5103
anind@cc.gatech.edu
http://www.cc.gatech.edu/fce/contexttoolkit

ABSTRACT
Context is an important, yet poorly understood and poorly utilized source of information in interactive computing. It will be of particular importance in the new millennium as users move away from their desktops and into settings where their contexts are changing rapidly. Context is difficult to use because, unlike other forms of user input, there is no common, reusable way to handle it. As a result, context-aware applications have been built in an *ad hoc* manner, making it difficult to build new applications or evolve existing ones. In this research, we are examining the requirements of context-aware applications, building a toolkit, which enables the use of context and fulfills these requirements, and testing the usability of this toolkit for application designers.

Keywords
Context, context-awareness, ubiquitous computing, application building

INTRODUCTION AND MOTIVATION
In human-human interaction, a great deal of information is conveyed without explicit communication. Gestures, facial expressions, relationship to other people and objects in the vicinity, and shared histories are all used as cues to assist in understanding the explicit communication. These shared cues, or *contex*t, help to facilitate grounding between participants in an interaction [2]. We define context to be any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object.

In human-computer interaction, there is very little shared context between the human and the computer. Context in human-computer interaction includes any relevant information about the entities in the interaction between the user and computer, including the user and computer themselves. Humans naturally provide context in the form of signs of frustration or confusion, for example, but computers cannot sense or use it. We define applications that use context to provide task-relevant information and/or services to a user to be *context-awar*e. For example, a context-aware tour guide may use the user's location and interests to display relevant information to the user [1].

In the past few years, there has been a tremendous increase in the use of handheld computers. In the next millennium, we will continue to see users move away from a traditional desktop-computing environment. Users will be increasingly mobile, using more handheld devices and interacting with computationally-enhanced environments, as we move towards Weiser's vision of ubiquitous computing [6]. In these mobile settings, a user's context, such as her location and the people and objects around her, is more dynamic. Both handheld and ubiquitous computing have given users the expectation that they can access information services whenever and wherever they are. With a wide range of possible user situations, we need to have a way for the services to adapt appropriately, taking into account the user's context, in order to best support the human–computer interaction.

CURRENT STATE OF CONTEXT-AWARE COMPUTING
Past research in context-aware computing has shown the value of using context in interactive applications. The use of context allows an application to be tailored to a user's specific situation [3,5], providing increased benefits to the user.

The increased availability of commercial, off-the-shelf sensing technologies is making it more viable to sense context in a variety of environments. The prevalence of powerful, networked computers makes it possible to use these technologies and distribute the context to multiple applications, in a somewhat ubiquitous fashion. So what has hindered applications from making greater use of context and from being context-aware?

A major problem has been the lack of uniform support for building and executing these types of applications. Most context-aware applications have been built in an *ad hoc* manner, heavily influenced by the underlying technology used to acquire the context. This results in a lack of generality, requiring each new application to be built from the ground up. To enable designers to more easily build context-aware applications, there needs to be architectural support that provides the general mechanisms required by context.

REQUIREMENTS OF CONTEXT
At first glance, one might suggest that we deal with context in the same manner that we deal with explicit user input such as keyboard and mouse activity in a graphical user interface (GUI). However, upon further investigation, we can identify three important distinctions between dealing with context and dealing with user input. These distinctions between context and user input drive the requirements for a

toolkit to support the development of context-aware applications.

The first distinction is that in traditional desktop computing applications, the source of user input is from a single computer, but with context-aware applications, the source of context is from many, distributed computers. Both user input and context require abstractions to separate the details of the sensing mechanisms from the use of the sensed information. A second distinction is that the use of context requires additional abstractions because context is often not in the form required by an application. For example, a keyboard provides the character-level input that an application requires. But, a global positioning system (GPS) provides location information in latitude-longitude format, whereas a context-aware application may require the name of the street the user is on. A third distinction is that GUI widgets that acquire user input belong to the application that instantiated them, whereas components that acquire context do not belong to an application, but instead execute independently of an application. This independence is required because context acquisition components must often provide context information to multiple context-aware applications running simultaneously.

CONTEXT TOOLKIT

We have built a toolkit in Java that addresses the distinctions between context and user input. The toolkit enables application developers to built context-aware applications. It consists of three main abstractions:

- **Widgets**. Just as GUI widgets mediate between a user and an application, context widgets mediate between a user and the environment [4]. Context widgets encapsulate information about a single piece of context, such as location or activity, for example. They provide a uniform interface to components or applications that use the context, hiding the details of the underlying context-sensing mechanisms.
- **Aggregators**. Context aggregators are typically used to aggregate context information of real world entities such as users or places. By acting as a gateway between applications and elementary widgets, aggregators hide even more complexity about the context-sensing mechanisms.
- **Interpreters**. A context interpreter is used to abstract or interpret low-level context information into higher level information. For example, identity, location, and sound level information could be used to interpret that a meeting is taking place.

The toolkit makes the distributed nature of context transparent to context-aware applications. Applications do not need to know whether these context components are being executed remotely or locally. These components run independently of any single application, allowing them to be used by multiple applications.

RESEARCH HYPOTHESES

As part of my research, we will use the context toolkit to evaluate the following hypotheses:

- Users prefer to use the widget, aggregator and interpreter abstractions over dealing directly with context sensors
- The context toolkit and design process allows application developers to more easily build and evolve context-aware applications
- Context solutions can be reused by other application developers

SUMMARY AND FUTURE RESEARCH

The goal of our research is to provide a toolkit that makes it easier for application developers to use context. The toolkit will enable developers to add context to applications that are not context-aware and to increase the use of context in applications that are already context-aware. We have completed the first version of our context toolkit and have given it to members of our research group as well as to other research institutes.

We have built a number of applications that demonstrate the usefulness of the context toolkit. We are currently attempting to identify a design process and a set of heuristics for building context-aware applications. We plan to study context-aware application developers to determine whether our hypotheses are correct. Our research contribution is in the area of applications design methodology.

ACKNOWLEDGMENTS

I would like to thank my advisor, Gregory Abowd, for his unending support and encouragement. This work was supported by a Motorola University Partner in Research fellowship.

REFERENCES

[1] Abowd, G.D. *et al*. Cyberguide: A mobile context-aware tour guide. *Baltzer/ACM Wireless Networks*, 1997, 3 (5), pp. 421-433.

[2] Clark, H.H. & Brennan, S.E. Grounding in communication. In L.B. Resnick, J. Levine, & S.D. Teasley (Eds.), *Perspectives on socially shared cognitio*n. Washington, D.C. 1991.

[3] Dey, A.K. *et al*. The Conference Assistant: Combining context-awareness with wearable computing. To appear in the Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99).

[4] Salber, D., Dey, A.K., and Abowd, G.D. The Context Toolkit: Aiding the development of context-enabled applications. *Proceedings of CHI'99*, 1999, pp. 434-441.

[5] Schilit, B., Adams, N. Want, R. Context-Aware Computing Applications. Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, 1994, pp. 85-90.

[6] Weiser, M. The Computer of the 21st Century. *Scientific American*, 1991, 265 (3), pp. 66-75.