Journal of Ambient Intelligence and Smart Environments 1 (2009) 57–62
DOI 10.3233/AIS-2009-0008
IOS Press

Modeling and intelligibility in ambient environments

Anind K. Dey
Human-Computer Interaction Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. E-mail: anind@cs.cmu.edu

Abstract. The field of ambient environments and ubiquitous computing has matured greatly over the past 20 years. We are no longer building toy applications but are focusing on real applications that have real impacts on user. We must now consider usability concerns when designing these applications, and particularly support for intelligibility, or the ability of applications to explain their behavior. This paper discusses the importance of intelligibility, particularly for applications that model human activity, and describe work my group has conducted in understanding how users understand these applications and toolkit support for intelligibility.

Keywords: Intelligibility, modeling context-aware, ambient environments

1. Introduction

Weiser's vision of a future world with ubiquitous, invisible computing [22] has led to the promise of smart environments that anticipate, adapt to, and provide for occupants' needs. An important aspect of this vision is context-aware computing, where applications adapt their behavior to changes in their local context [17]. 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 application themselves [7]. Application behavior adaptation can include presenting information related to the current context, presenting a list of services relevant to the current context and executing a service based on the current context. A common example of a context-aware application is the mobile tour guide [1]. A user walks around a smart environment with a handheld device that proactively shows information to the user about the exhibit or venue she is in front of. In addition, it can direct the user to other exhibits that are similar to the ones the user has enjoyed most, based on the amount of time she has spent at each exhibit, for example.

The vast majority of the research in context-aware computing in smart environments has been focused on two main areas. The first area is the building of novel applications, such as the tour guide systems mentioned above, reminder systems [6], environmental control systems [8], location-based services [21] and context-based retrieval systems [9] where stored information is tagged with context to aid later retrieval of that information. The second area is the building of software infrastructure, or middleware, that supports programmers in more easily building context-aware applications [3,7,18]. These efforts are an important first step towards context-awareness being viable, and now that it is possible to build interesting applications, there is a need to address the challenges faced by end-users – the occupants of smart environments – who are being asked to adopt and use these applications.

Similarly, a major focus in ubiquitous computing is in recognizing human activity in these smart environments. The recent Ubicomp, Pervasive Computing and Ambient Intelligence conferences are full of papers recognizing a variety of human activities such as medicine taking [20], movement through a house [16], and washing dishes [12]. The ability to model everyday human behavior and to accurately infer user

intent and goals from that behavior is the *holy grail* of context-aware computing. While the focus of context-aware computing has been on location and other simple forms of context, these are just proxies for user intent.

However, the ability to accurately model this behavior is limited at this time to simple activities and intent cannot be adequately inferred. What often results are either simplistic applications that have been scoped down so much that they are not interesting, or interesting applications that often make mistakes in selecting and executing a context-aware behavior. As a research community developing and evaluating ambient intelligent systems for smart environments, we should be focusing on pushing the boundary of what can be built, and building more interesting applications. That means needing to both improve our ability to model real human activities, and provide support to users when applications do things that are confusing or are incorrect.

2. Activity modeling

My research group in the Human-Computer Interaction Institute at Carnegie Mellon University is actively engaged in modeling real human behavior[1]. Currently, four different types of activities are being modeled:

- how users, and in particular elders, drive [23];
- how and when users participate in physical activity [11];
- the routines and deviations of dual-income families [4]; and,
- how users with memory impairments engage with and retain memories [10].

In each of these settings, it is currently impossible to perfectly model these activities. As with any real human behavior, there is a lot of randomness in sensed data about these behaviors and it is not possible to sense everything you want to about a particular behavior. In the absence of this perfection, my group is investigating the concept of *intelligibility*, which is a system's ability to explain its own behavior, both the suggestions it makes or the actions it takes and the rationale or reasoning process for coming up with those suggestions and actions.

[1] More information on these projects available at http://www.cs.cmu.edu/~anind.

3. Intelligibility

One of the biggest challenges to the usability of context-aware applications is the difficulty of understanding why they do what they do. The lack of intelligibility of an application can be a serious issue that may lead to application failure or abandonment due to lack of trust [14] in the system. Intelligibility as an application feature includes supporting users in understanding, or developing correct mental models of what a system is doing, providing explanations of why the system is taking a particular action, and supporting users in predicting how the system might respond to a particular input. Mental models are a hypothetical construct defined as a mental representation of a real or imagined situation. This information is essential for helping users form useful mental models about systems and making systems usable [15]. Without this information, users may even abandon or refuse to adopt a system because they are unable to understand how it works, or even whether it is working. In other related fields, such as intelligent agents, researchers have established that the two major design issues for such systems are accuracy/performance and user trust [13]. For example, the Clippy Microsoft Agent has largely been abandoned because it frustrated users by making erroneous suggestions with no explanation of why these suggestions were being made. In response to user frustration, Amazon.com added a link under a user's recommendations: "Why is this recommended for you?" In human-computer interaction (HCI), feedback is a fundamental feature of any desktop application, and, in general, HCI researchers and practitioners know how to support intelligibility for this domain. However, there are significant challenges when working with context-aware applications and supporting intelligibility has been largely ignored in this domain [5].

First, context-aware applications are typically designed in two ways: either as a collection of deterministic if-then rules where a rule is fired when the prespecified collection of context has been observed, or as a machine learning system where context is input into a learned model and an action is probabilistically chosen. In the first case, when the collection of rules is large enough and there is overlap in the rule base, it can be quite difficult for a user to understand why a particular rule was fired or what she needs to do in order to obtain a particular action. In the second case, when a machine learning system is being used, because the system may change its behavior over time,

Fig. 1. Door displays outside managers' offices.

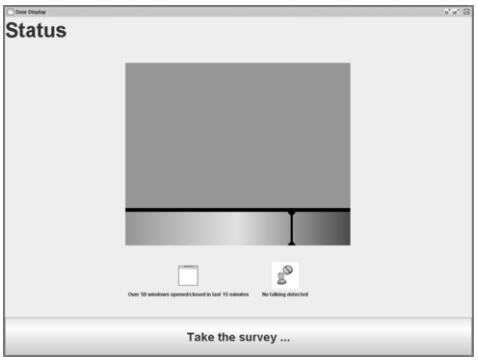

Fig. 2. Door display visualization to support intelligibility.

its behavior may be quite difficult to understand and predict. Clippy, spam filters and recommendation systems are good examples of this latter case. It is clear that the issues of rules and machine learning systems apply to more than just context-aware applications, but they are the norm in context-aware systems.

Second, and more specific to context-aware applications, is that they use mostly implicit input to determine what actions to take on behalf of a user. Users may have little understanding of what the system considers to be input, unlike in a desktop application where input is explicitly specified. With the input to context-aware systems being implicit, understanding why an application took a particular action or predicting what action the system will take given particular user action, can be very difficult [5]. As a simple example, take a context-aware audio tour guide that provides an increasing amount of information as the user shows more engagement with an exhibit. If the system inferring engagement measures (and these systems usually do) how much time the user is in front of the exhibit, it could present information to a user even though he may just be having a conversation with a friend near the exhibit. Without understanding the model of what the system is sensing and inferring, and how it uses this information, users can get quite frustrated.

As stated above, intelligibility derives from a combination of users' mental models of, explanations of, and predictions about application behavior. From the perspective of an application designer, this means that it is crucial to understand how mental models develop, what sorts of explanations of a system will support correct mental model development, and what feedback will allow users to make correct predictions.

4. Understanding how mental models develop

An initial exploration has been performed about how users form mental models about complex systems, within the domain of interruptibility. In a 6-week pilot study of how non-technical users form mental models, a context-aware system was deployed that used user-trained sensor-based statistical models to provide estimates of an individual's interruptibility [19]. These estimates were displayed outside the individual's door for coworkers to see (see Fig. 1). Two separate deployments were conducted, one in which 6 co-workers only saw the interruptibility estimate, and one in which a different set of 6 co-workers saw the interruptibility estimate along with up to 3 features that contributed the most to the interruptibility estimate (*e.g.*, talking detected, typing on the computer and changes in window focus) (see Fig. 2). During the deployment, co-workers who interacted with these displays were interviewed, eliciting their mental model of how the system was working. The interviews focused on soliciting the set of sensors subjects thought were contributing to the estimate, the relative importance of sensor inputs, how these inputs were synthesized into an estimate and finally, how their experiences in using the system helped to confirm their understanding of the system.

Subjects reported fewer numbers of sensors being part of the system as the study progressed. In terms of correctness of sensors reported, subjects also improved as time went on, with a higher percentage of sensors reported actually being part of the real system. In both cases, however, there was no difference between those who received feedback about relevant features and those who did not.

Subjects also reported a wide variety of model types when describing how the interruptibility

estimate was arrived at. Two subjects described a simple set of rules (*e.g.*, if user is on phone, set estimate to highly uninterruptible). One subject believed that the display was being controlled manually by the user whose interruptibility was being estimated. Two subjects described a decision tree-like model in which the system builds a list of activities from past history and tests the current situation along a number of branching conditions to see if it matches an activity in the list. The final three subjects described a similar model, with the addition of simple statistics to determine degrees of interruptibility (*e.g.*, deviation from mean level of historical sensed activity).

While these three subjects were able to describe a model that was closest to the actual system, incorporating history and statistics along with prioritized sensors, our other five subjects were not. To be understandable to users, context-aware systems that use machine learning and estimates of current state must effectively communicate the key concepts of learning from history and statistical inference from current sensor data. Equally clear from our results is that incorporating feedback about relevant features was of only moderate use in helping subjects understand the system's operation. Similarly, the use of a gradient scale to represent interruptibility estimates was only successful in conveying the continuous nature of the estimate to half of our subjects, and fewer made the connection between this and statistical patterns.

It is quite surprising that the subjects basically described the same model structure throughout the study. The subjects were expected to frequently modify their models as they interacted with the system more over time and observed its behavior. Even when they had conflicting experiences or clear deficiencies in incorporating known components of the system, they maintained their earlier model. Moray's theory of "cognitive lockup", where operators maintain their beliefs about a system even in the face of contradictory evidence is intended to be applied to expert users who have an understanding of the system's normal operating parameters, but it appears that it applies to relatively novice subjects as well.

An interesting final result was that participants were able to ascribe basic machine learning concepts (*e.g.*, decision trees, statistical analysis) to our context-aware system, despite being unfamiliar with the field. These results are encouraging in that designers can potentially know what to expect in terms of user sophistication with respect to machine learning.

5. Toolkit support for intelligibility

While more studies need to be conducted into how to best support intelligibility in context-aware applications, the next step is to build support for intelligibility into a toolkit for context-aware applications, To this end, the Context Toolkit [7] has been augmented with such capabilities, at least initial ones.

With existing context infrastructures, a simple smart environment application such as automatically controlling the lighting in a home based on location of people would be implemented as follows: The application logic consists of finding a discoverer, querying it for relevant people inputs, and subscribing and maintaining connections to each of those inputs. When the application receives data from each input, it must maintain internal state information keeping track of each person's location. When a user's location matched a location of interest, an output or service would be called to change that location's lighting appropriately.

The Context Toolkit was augmented with a new abstraction, Situation, to simplify the development of such applications. Now, the application logic consists of creating a Situation with a description of the information it is interested in (locations of specific people) and what the Situation should do when (set the lighting level based on the occupancy). The application logic would consist of a number of context rules of the following form: when any of users (A, B, C, D) are in the kitchen and it is nighttime, turn on the overhead lights to maximum. The Situation handles the remaining logic: discovery, mapping to individual sources of context and data, determining when input is relevant and executing appropriate services. Developers can easily encapsulate their context-aware logic using Situations. Situations also provide the necessary functionality to obtain a real-time execution trace for an application. They have methods for exposing the context rules they encode and for changing the values of parameters.

Situations were extended to include an *Introl* object, to support *in*telligibility and con*trol*. It is usually invisible, but can be made visible either through the application user interface or via a keyboard hotkey. Introl provides information on context rules and their execution and current context values (as well as limited information on how they were derived), and the ability to modify parameters (and therefore application logic). While this is likely too "clunky" for most

Fig. 3. Activity Monitoring intelligibility interface.

end-users, it does support basic intelligibility and control for end-users across all Situation-based applications with no extra effort required by the application developer. Building usable intelligibility and control interfaces is an open challenge [2] that the toolkit support for interface designers attempts to address.

To support interface designers, support for Situations was extended with language specific support for Visual Basic and Flash, two languages commonly used by designers. These extensions allow designers to create custom intelligibility interfaces based on the information that can be collected from Situation and Introl objects, and provide a tremendous amount of flexibility in designing these interfaces, including replacing the application's original interface. Figure 3 shows an example application built in Flash by a designer: an office activity monitoring application, in which a user can get more information about a user's state by hovering the mouse over the user's name.

6. Summary

Intelligibility is a crucial usability consideration in smart environments. While only a preliminary investigation has been conducted into how users form mental models about complex context-aware systems, it has resulted in some very useful insights about users' abilities to reason about complex systems, and the challenges in conveying how complex systems work. Additional studies are being planned to investigate how best to support intelligibility in context-aware systems. Some initial work has been conducted on toolkit support for building intelligibility interfaces. The overall approach is to conduct studies, explore the space of interaction techniques for improving intelligibility and then continue building reusable support for intelligibility into a toolkit.

Through this research, contributions will be made to the growing interest in context-aware and smart environment systems. This work will enable and empower users to engage with ambient intelligence systems by making them more understandable. This work will benefit smart environment researchers working in interaction design, application design and infrastructure support, as well as researchers looking to support intelligibility in complex systems, and make Ubicomp applications more useful and usable.

References

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

[2] M. Assad, D. Carmichael, J. Kay and B. Kummerfield. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. *Proc. Pervasive 2007* (2007), 55–72.

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

[4] V. Bellotti and W.K. Edwards. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction* **16**(2-4) (2001), 193–212.

[5] S. Davidoff, M.K. Lee, C. Yiu, J. Zimmerman and A. K. Dey. Principles of smart home control. *Proc. Ubicomp 2006* (2006), 19–34.

[6] A.K. Dey and G.D. Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. *Proc. HUC 2000* (2000), 172–186.

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

[8] S. Elrod, G. Hall, R. Costanza, M. Dixon and J. Des Rivières. Responsive office environments. *Communications of the ACM* **36**(7) (1993), 84–85.

[9] M. Lamming and M. Flynn. Forget-me-not: IntimateComputing in Support of Human Memory. *Proc. FRIEND21 '94* (1994), 125–128.

[10] M.L. Lee and A.K. Dey. Lifelogging memory appliance people with episodic memory impairment. *Proc. Ubicomp 2008* (2008), 44–53.

[11] I.A. Li, A.K. Dey and J. Forlizzi. Monitoring and feedback to increase awareness of exercise activities. Ubicomp 2005 Workshop on Monitoring, Measuring and Motivating Exercise: Ubiquitous Computing to Support Physical Fitness. 2005.

[12] B. Logan, M. Healey, E.M. Tapia and S. Intille. A long-term evaluation of sensing modalities for activity recognition. *Proc. Ubicomp 2007* (2007), 483–500.

[13] P. Maes. Agents that reduce Work and Information Overload. *Communications of the ACM* **37** (1994), 30–40.

[14] B. Muir. Trust in automation: Part I: Theoretical issues in the study of trust and human intervention in automated systems. *Ergonomics* **37**(11) (1994), 1905–1922.

[15] D.A. Norman. *The Design of Everyday Things*. New York, Doubleday. 1990.

[16] S.N. Patel, M.S. Reynolds and G.D. Abowd. Detecting human movement by differential air pressure sensing in HVAC system ductwork: An exploration in infrastructure mediated sensing. *Proc. Pervasive 2008* (2008), 1–18.
[17] B.N. Schilit, N.I. Adams and R. Want. Context-aware computing applications. *1st International Workshop on Mobile Computing Systems and Applications* (1994), 85–90.
[18] B.N. Schilit. *System architecture for context-aware mobile computing*. Ph.D. Dissertation, Columbia University, 1995.
[19] J. Tullio, A.K. Dey, J. Chalecki and J. Fogarty. How it works: a field study of non-technical users interacting with an intelligent system. *Proc. CHI 2007* (2007), 31–40.
[20] S. Vergun, M. Philipose and M. Pavel. A statistical reasoning system for medication prompting. *Proc. Ubicomp 2007* (2007), 1–18.
[21] R. Want, A. Hopper, V. Falcao and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems* **10** (January 1992), 91–102.
[22] M. Weiser. The computer for the 21st century. *Scientific American* **265**(3) (1991), 94–104.
[23] B.D. Ziebart, A. Maas, A.K. Dey and J.A. Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. *Proc. Ubicomp 2008* (2008), 322–331.