

# Intelligibility and Accountability: Human Considerations in Context-Aware Systems

**Victoria Bellotti** and **Keith Edwards**  
*Xerox Palo Alto Research Center*

---

## ABSTRACT

This essay considers the problem of defining the context that context-aware systems should pay attention to from a human perspective. In particular, we argue that there are human aspects of context that cannot be sensed or even inferred by technological means, so context-aware systems cannot be designed simply to act on our behalf. Rather, they will have to be able to defer to users in an efficient and nonobtrusive fashion. Our point is particularly relevant for systems that are constructed such that applications are architecturally isolated from the sensing and inferencing that governs their behavior. We propose a design framework that is intended to guide thinking about accommodating human aspects of context. This framework presents four design principles that support *intelligibility* of system behavior and *accountability* of human users and a number of human-salient details of context that must be accounted for in context-aware system design.

---

**Victoria Bellotti** is a senior research scientist in the Computer Science Laboratory at the Xerox Palo Alto Research Center, with interests in ethnography and computer-supported cooperative work. **Keith Edwards** is computer scientist with interests in collaboration, distributed object systems, and novel user interfaces; he is a senior research scientist in the Computer Science Laboratory at the Xerox Palo Alto Research Center.

---

## CONTENTS

- 1. INTRODUCTION**
  - 2. HUMAN CONSIDERATIONS IN CONTEXT**
    - 2.1. Basic Context Awareness: Responsiveness to the Environment
    - 2.2. Human Aspects of Context: Responsiveness to People
    - 2.3. Social Aspects of Context: Responsiveness to the Interpersonal
    - 2.4. Summary
  - 3. A FRAMEWORK FOR INTELLIGIBILITY AND ACCOUNTABILITY IN CONTEXT-AWARE SYSTEMS**
    - 3.1. Four Design Principles and Human-Salient Details Required to Realize Them
      - Context-Aware Principles to Support Intelligibility and Accountability
      - Human Salient Details of Context
    - 3.2. Applying the Principles
      - Inform the User of Current Contextual System Capabilities and Understandings
      - Provide Feedback
      - Enforce Identity and Action Disclosure
      - Provide User Control
  - 4. DISCUSSION AND CONCLUSIONS**
- 

## 1. INTRODUCTION

Imagine you are in a context-aware building recording verbal input to some document. Someone elsewhere in the building requests that the infrastructure locate you and establish a communications channel. What should your context-aware system do? Should it disclose your presence, location, and status information, such as that you are working but available? Should it just make the connection, or identify the caller and request your permission to connect? Should it determine that you are occupied and send a reply to the caller that you are unavailable?

The answers to these questions and many others will partially depend on the nature of the system; is it an audio-only communications infrastructure or an audio-video informal knowledge-capture system, documenting all your conversations? They may depend on whether you are working in your bedroom or a public room such as a library, and on whether you are working on your resignation letter or giving a lecture to a room full of people. Furthermore, they may depend on your relationship with the person trying to contact you, your mood, or what you are wearing (a robe, a smart suit, or nothing at all).

In fact, the appropriate system action might depend on myriad factors that cannot be reliably sensed or inferred. The only entity that can ultimately decide what to do is you. The only way you can decide is if you understand the nature of the context-aware application, what it proposes to do for you, and whom it is attempting to connect you to.

The anchor article by Dey, Abowd, and Salber (2001 [this special issue]) is not just about designing context-aware applications, but also proposes a conceptual framework that makes the design of such systems simpler with a component-based approach to system architecture. Implicit in their definition of context awareness is the assumption that, by being “aware” of the “context” a system can infer that it should take certain actions (e.g., by invoking services such as sending e-mail or rerouting audio to another room). However, their component-based framework separates the applications that take actions from the underlying context sensing and inferencing architecture that predetermines the inputs that cause those actions.

For these reasons, we propose that the framework needs to be expanded to handle the variability highlighted in our opening example. Like other authors in this special issue, we believe that context is a problematic and contingent idea; it is, perhaps, whatever might be considered relevant to the current act from sleeping to printing a document. It might include anything a system can sense (e.g., CO<sub>2</sub> levels, lighting levels, speech, movement, devices, systems, locations) and anything that can be inferred (e.g., identity, role, relationship, interest, intent, conversation); the lists go on and on.

Although it is difficult enough to make reliable inferences about the nature and status of environmental conditions and systems, it is the human and social aspects of context that seem to raise the most vexing questions. Although these are the very aspects of context that are difficult or impossible to codify or represent in a structured way, they are, in fact, crucial to making a context-aware system a benefit rather than a hindrance or—even worse—an annoyance.

In this essay we take issue with two problems that are implicit in the assumptions of the framework proposed by Dey et al.

First, designers are unlikely to be successful at representing human and social aspects of context in a deterministic fashion, such that machines can take autonomous action on our behalf. In the absence of the ability of a system to make meaningful inferences about human context, we argue that, instead, a set of design principles are required that enable human beings to reason for themselves about the nature of their systems and environments, empowering them to decide how best to proceed. This entails making certain contextual details and system inferences visible to users in a principled manner and providing effective means of controlling possible system actions.

Second, a component-based approach to building context-aware systems may not properly equip the designer with the tools to provide the visibility and

control mentioned earlier. In a component-based approach, designers are largely restricted to creating application code that uses existing components, if they are to achieve the reuse that is the *raison d'être* of such approaches. However, designers will often need to reach beyond the application to refine or augment other components to deliver capabilities not anticipated by the original component builders.

The rest of this essay explains these propositions in more detail and presents a framework for designing context-aware systems that respect human initiative, rather than attempting to supplement it. We argue that this framework is complementary to that proposed by Dey et al. and deals with some missing fundamentals that are required to build usable context-aware systems.

## 2. HUMAN CONSIDERATIONS IN CONTEXT

In this section we attempt to explain in more detail why people are so difficult to deal with as contextual entities. We argue that it is because people, unlike systems and devices, make unpredictable judgments about context. In other words, they *improvise* (Suchman, 1987). This is because they have intentions, emotions, dislikes, phobias, perceptions, interpretations (and misinterpretations) and many other motivators that drive their behavior in unpredictable ways that are impossible to even model accurately, let alone instrument or infer. These motivations drive the initiative that people take when they interact with the world about them and with other people and they are essentially contingent and nondeterministic. Further, experience has shown that people are very poor at remembering to update system representations of their own state; even if it is something as static as whether they will allow attempts at connection in general from some person (Bellotti, 1997; Bellotti & Sellen, 1993) or, more dynamically, current availability levels (Wax, 1996). So we cannot rely on users to continually provide this information explicitly.

Thus, only the basic nonhuman aspects of context, with constrained conditions and well-defined, responsive behaviors, can be handled by devices on their own. In the following subsections we illustrate with some familiar examples how things, in context-aware systems, become increasingly complicated as people, going about their unpredictable daily business, are added to the equation.

### 2.1. Basic Context Awareness: Responsiveness to the Environment

We already live in a world full of basic context-aware systems that respond to their environment in some way: air conditioners; security systems with sound and motion detectors and CO sensors; assembly plants with safety sen-

sors; passenger and military aircraft that sense their speed, height, pitch, and so forth and can essentially fly themselves. However, such systems can only supplant human initiative in the most carefully proscribed situations. The more unpredictable the operating conditions, the more sensing is required to make the system aware of what it needs to know to operate, and, in turn, the more difficult and complicated it becomes to an engineer such that it will behave correctly within its operating constraints. For example, it took 10,000 people in 238 cross-functional “design build teams” 4 years to complete the Boeing 777 (Boeing, 1995).

## **2.2. Human Aspects of Context: Responsiveness to People**

Context-aware systems, as we tend to define them in the HCI research community (though one might reasonably call the auto-pilot of a Boeing 777 context aware), are expected to be responsive to more generalized situations where people become a much more significant part of the context (Abowd et al., 1997; Dey, Futakawa, Salber, & Abowd, 1999; Schilit, Adams, & Want, 1994). This kind of investigation is interesting and may yield useful results in some circumstances, even outside of the artificial laboratory settings where the research is being undertaken. But, in real situations, there are genuine risks involved in allowing the system to take the initiative in any activity in which human participants are involved (Rochelin, 1997). As we see, there is simply too much variability as to what should be done, given different conditions, for designers to successfully model appropriate outcomes in advance.

Dey et al. go beyond the effort to simply build context-aware systems to claim that context-aware applications can be architecturally separated from the components that manage inference, and the actions that may result. These arguments are based on traditional concerns of systems developers and a component-based approach is indeed a useful way of simplifying the engineering of any system. But it is also important not to throw the baby out with the bath water, so to speak. The “context widgets,” “interpreters,” and “aggregators” proposed in Dey et al.’s framework are separate applications from their own reasoning, rendering them ignorant of the means by which they acquire their representation of context. This may not be problematic when the output of the application simply represents the location of a sensed object. Things get much more complicated when applications take it upon themselves to interpret human activity and “do things” for people by invoking services (which may also be separate, prefabricated components in Dey et al.’s framework).

In most situations where context-aware systems have been proposed or prototyped (meeting rooms, offices, homes, and conferences are typical examples in the scenarios we see in the literature), human initiative is frequently required to determine what to do next. For example, a context-aware

application can measure temperature, but it cannot tell when a room is too hot and needs to be cooled. That determination must be left to the users. This is, of course, why heaters and air conditioners have accessible thermostats, generally in a central location, which can easily be tweaked on a regular basis. Some people like to keep their home very warm, others prefer it cool, and perhaps, if there is more than one occupant, a compromise will be struck, or one will keep turning the heat up while another keeps turning it down. If a single homeowner has just come back from a jog, her regular setting may feel too hot. There is no reliable way for an air conditioning system to make these kinds of determinations on home occupants' behalf. Thus people must alter the setting themselves.

Not only does the user have direct control, allowing them to raise or lower the preferred temperature set-point, but a typical thermostat will also display the approximate temperature it is trying to maintain. If the room where the thermostat is located seems very hot but the air conditioner is not succeeding in cooling the room, even though it is showing a reasonable temperature setting, the user can deduce that there is something wrong with the system. This is because they know something about how the system works that, in turn, arises because the system is in some way intelligible. This is a key requirement of context-aware systems, no matter how simple.

Imagine if the thermostat were in some unknown location, or did not display the current set temperature, or the user could not tell whether he had set the thermostat to a desirable temperature, or if the user did not even know that a thermostat existed or how it behaved. The system might function perfectly well from a technical standpoint, but it would be much harder to operate from a user perspective. Information that is necessary for the user to get the system to work for him or her would no longer be available. This is roughly the effect one risks by building a context-aware application that is ignorant of and thus hides the nature of its own sensing and inferencing, as proposed by Dey et al.

In short, systems cannot just do things based on context awareness; rather, we argue that they are going to have to involve users in action outcomes if they are to be acceptable. But to judge whether a system should do something, a user needs to understand what the system is doing, or about to do, and why it thinks it should. Separating applications from the basic sensing, inferencing, and service components, whatever they may be, allows users to interpret a context and its potentials and risks making the human task of controlling a system's action or responding to a suggestion for action much more difficult. Designers must not be lulled into a false sense of security that these aspects of their design are "already taken care of" and do not require scrutiny and possible refinement.

### 2.3. Social Aspects of Context: Responsiveness to the Interpersonal

Context-aware systems can identify co-location, and perhaps conversation, but they cannot distinguish between a speech and an inappropriately emotional outburst. They can model standard information-sharing characteristics of a formal relationship between a manager and her subordinate assistant, but they cannot sense whether the manager trusts the assistant and what exactly she trusts her with, and vice versa.

In fact, the more we try to get systems to act on our behalf, especially in relation to other people, the more we have to watch every move they make. For example, it has often been proposed that people might like to share their notes at a meeting or conference (Abowd et al., 1998; Dey et al., 2001 [this special issue]). However, in some previously unreported research that we have undertaken on personal information management (other results of this work were reported in Bellotti & Smith, 2000), we found that many people are not really comfortable with sharing their notes. In practice, it may be necessary to make distinctions between people who can and cannot be allowed access to one's notes, and when. Furthermore, one might well be interested to know when and how often one's notes are viewed or copied by any particular individual.

In a thought-provoking, related example, a system in use at Apple Computer® (Cohen, 1994) conveyed, through auditory feedback, a sense of when public folders on one's hard drive were being accessed by other users. Some people were surprised and quite disturbed to discover how often their voluntarily shared folders were accessed by unknown others. Indeed, some even felt invaded (Cohen, personal communication, 1996). The idea of disembodied visitors arriving unheralded, rummaging around on one's hard-drive, and looking at who-knows-what turns out to be quite unsettling.

Thus, mediation between people is an ambitious and potentially threatening aim for context-aware systems. What might it be like to live in a world where personal information becomes available as one moves from one space to another? It is hard enough to keep track of shared files on a desktop PC, but, with new context-aware systems, how will we know when information is captured, accessed, and used, and by whom, for what purposes in context-aware settings? And how will this kind of capability make us feel? Authors in the field of computers and privacy have already registered concerns about the extent to which existing networked systems can capture information about our activities and make it available for ends that conflict with our own (Agre & Rotenberg, 1997; Dunlop & Kling, 1991; Neumann, 1995). So, looking forward, as more and more of our data and actions become accessible to systems as they become more aware, what measures can we take to ensure that we are aware of the implications of this and are also able to deal with them?

## 2.4. Summary

Let us summarize our arguments so far:

- Context-aware systems are here. Context-aware systems already exist, but their commercial application is mainly restricted to sensing and responding to physical or systems states and events.
- Context-aware systems infer human intent. New context-aware systems have been proposed that seek to determine, by means of inference from what can be sensed, that some human intent exists that can be represented or served by some system action. For example, if I wave my device in front of a tagged poster or document, the system may download relevant information or a copy of the document to my device (Want, Fishkin, Gujar, & Harrison, 1999).
- Context-aware systems mediate between people. Going beyond inferring human intent, systems are being designed to interpret interpersonal relations and events. For example, a system can infer that people who are co-located might wish to share information or documents (Dey et al., 2001 [this special issue]; Edwards, 1994). And if someone moves from office to office, such a system might redirect his phone calls, inferring that he wants his calls to follow him around the building (Want et al., 1992).
- Context-aware systems must be accountable and so must their users. Users need to be able to understand how a system is interpreting the state of the world. Context-aware systems must be *intelligible* as to their states, “beliefs,” and “initiatives” if users are to be able to govern their behavior successfully (Dourish, 1997). Further, because there is a tendency for computer mediation between people to bring about problems of disembodiment (Bellotti & Sellen, 1993; Heath & Luff, 1991), context-aware systems must also provide mechanisms that enforce accountability of users to each other.
- There is a danger inherent in component-based architectural separation of sense and inference from system initiative. If applications are ignorant of their own inferences, they cannot give any useful account that will help users to judge the best course of action in their use (as discussed in our air conditioning example). This will be especially problematic when systems begin to mediate between multiple users who need to be sensitive to each other’s potentially unpredictable intentions and conflicting interests. If we do not take this issue into account in our designs, we can expect to see problems, similar to those that inspired the implementation of caller-ID in our telecommunications systems.

Where, then, do these arguments leave us? Foremost, we believe that complex machine inferencing based on human context is a difficult proposition, especially when the system attempts to take action based on a presumption of human intent; this is a case also made by others in this issue (e.g., Greenberg, this special issue). This is not to say that inferencing can't be sometimes useful. But even a simple inference, when done at all, must be situated within a particular application. It is applications that embody particular domain semantics, and also have the architectural "proximity" to the user to be able to defer to him or her when an interpretation is ambiguous. If an application is architecturally separated from its own inferencing, this issue becomes all the more challenging.

In the absence, then, of the ability to off-load onto the system the responsibility for taking action based on the presumption of our intent, a set of design principles are needed that will enable users to be able to reasonably and rationally make their *own* inferences based on the state of the world as perceived by them, and as reported by the system. We propose that two key features must be supported in any context-aware infrastructure to allow users to make informed decisions based on context:

- **Intelligibility:** Context-aware systems that seek to act upon what they infer about the context must be able to represent to their users what they know, how they know it, and what they are doing about it.
- **Accountability:** Context-aware systems must enforce user accountability when, based on their inferences about the social context, they seek to mediate user actions that impact others.

To realize these two features, context-aware systems must embody certain design principles and support communication of some key, human-salient information. These guidelines describe systems that don't *require* complex inferencing to be effective, or—if inferencing does take place—these guidelines acknowledge the fallibility and inaccuracies inherent in such inferencing by equipping users with the tools necessary to interact intelligently and responsibly with context-aware systems. To provide these tools, designers may need to reach beyond the implementation of a particular application itself and design additional resources into existing components such as widgets, interpreters, aggregators, and other services. In the remainder of this essay, we present these guidelines as a design framework to compliment that of Dey et al. for the design of context-aware systems.

### 3. A FRAMEWORK FOR INTELLIGIBILITY AND ACCOUNTABILITY IN CONTEXT-AWARE SYSTEMS

The following framework is composed of a series of principles that must be maintained to allow intelligibility and accountability and includes a listing of human-salient details that users are likely to need information about in context-aware settings.

#### 3.1. Four Design Principles and Human-Salient Details Required to Realize Them

The following principles are not hard and fast rules. There will always be situations where they might not apply. However, we argue that designers must explicitly rule them out as unnecessary, or requiring too great a trade-off against some other design factor such as response time, as opposed to not considering them at all.

#### Context-Aware Principles to Support Intelligibility and Accountability

1. *Inform* the user of current contextual system *capabilities and understandings*.
2. Provide *feedback* including:
  - Feedforward: What will happen if I do this?
  - Confirmation: What am I doing and what have I done?
3. Enforce *identity and action disclosure* particularly with sharing nonpublic (restricted) information: Who is that, what are they doing, and what have they done?
4. Provide *control* (and defer) to the user, over system and other user actions that impact him or her, especially in cases of conflict of interest.

Realizing these principles depends on the context-aware system infrastructure being able to model human-salient details of the context, based on technically sensed events. These are listed next:

#### Human-Salient Details of Context

- Identity of others and self within and enabled by capabilities of the context. This might include some definition of role or interpersonal relationship that implies certain capabilities within some situation or with respect to others in that situation. This also includes control over

your own self-representation—what information you give out about your identity.

- Arrival of others or of oneself in a context, or *commencement* of a context.
- Presence of others and oneself within a context. This might include information about location and capabilities.
- Departure of others or oneself from a context, or *termination* of a context.
- Status and availability of one's own and others' actions or data to the context. This might usefully include an indication of *activity* and might also include representation of some *abstractions of status* such as “participating,” “busy,” “out,” “in transit,” or “waiting” and *abstractions of availability* such as “public,” “shared,” “restricted,” or “private” that imply intentions or potentials within the context. However, requiring the user to constantly update this information is unacceptable.
- Capture of information by the context (e.g., video, identity, location, action, etc.).
- Construction of information by the context (i.e., in Dey et al.'s terms, how data might be interpreted, aggregated, and stored).
- Access of information (by systems or people) from the context.
- Purpose: what use is made of information from the context (by systems or people; e.g., viewing, copying, modification, and, in particular, persistence or *storage*).
- Type of situation: information about the governing *social rules* of the context and the *technical implementations* of those rules that follow from the situation. Familiar constructs such as “lecture,” “meeting,” “library,” “exhibition,” and “interview” may be useful as metaphors for encapsulating a set of rules.

These principles and socially salient details of context represent the evolution of a framework originally proposed in Bellotti and Sellen (1993), which was intended to inform design for privacy in ubiquitous computing environments (which equally applies to context-aware environments). However, the framework is developed here to facilitate intelligibility of pure system behavior on behalf of users as well as system-mediated human behavior that might impact users' sense of privacy.

### 3.2. Applying the Principles

In this section we attempt to illustrate why and how the principles we propose should be embodied within the design of context-aware systems. In doing so we expose the importance of presenting human-salient details to system users so that they may operate *effectively*, *responsibly*, and *confidently* with such systems.

## **Inform the User of Current Contextual System Capabilities and Understandings**

We know that people are highly sensitive to their social context. Goffman (1959, 1963) discussed the many ways in which people modify their own behavior (performance) subject to context (region) in order to present a credible and respectable front to others present. A shared understanding of the nature of the setting and what it implies is necessary for contextual and interpersonal intelligibility and smooth interaction. Goffman (1963) wrote, “in ordinary life there is an expectation that all situated activity, if not obviously ‘occasioned,’ will have a degree of transparency, a degree of immediate understandability, for all persons present” (p. 76).

To interpret the behavior of others and to behave appropriately, participants must have knowledge of the *type of situation* they are participating in. We propose that designers ensure that context-aware systems are intelligible in this way. Metaphors can be a convenient shorthand for setting expectations appropriately. For example, a situation defined as a meeting might imply somewhat different sharing capabilities from an interview. A library setting might imply different information access capabilities from a private office setting. However, metaphors that imply traditional situations might be misleading when those situations become context aware.

Although Dey et al. agree that users need information about context, they treat the system itself as a nonproblematic and faithful mediator of that context. In fact, the systems they discuss may both intervene in and (depending on their limitations) misinterpret the context at various times (e.g., by inferring that colleagues want to share their notes in a ‘conference’ situation and publishing them automatically). Yet the framework proposed by Dey et al. provides no guarantees that the systems designed in accordance with it will offer sufficient information for users to effectively comprehend and respond to the situation they are in (as they could when they usually negotiate the sharing of notes at a conference). Such guarantees must be explicitly built in by the application designer, which may at times mean reaching into the logic of widgets, interpreters, aggregators, services, and discoverers to deliver explanations of system state and potentials (and, of course, appropriate control over these aspects; see later). For example, in a conference setting, users might wish to know if their application (via context widgets and interpreters) is using line-of-sight (infrared communication) or proximity (radio frequency communication) as the basis for initializing a notes-sharing session.

With respect to situation type, it is worth underlining the fact that human-salient contexts or situations may blur or subtly shift in ways that a so-called context-aware system cannot infer. They do not necessarily align well with such things as schedules, location, or architecture. Thus systems can

never be truly context aware in the way we might like them to be. However, because we are contemplating systems that, in some way, define a kind of *technical context*, we do need to be able to convey something of what this entails to their users. Goffman (1959) described how tangible regions play a key role in shaping appropriate behavior, but with context-aware infrastructures we risk creating intangible regions with obscure potentials and hazards that are likely to be difficult for people to perceive.

Thus, our framework argues for explicit additional cues about these technically enabled regions, which are not explicitly addressed by Dey et al.'s framework. For example, when my mobile device is somehow picked up by a system, I have in some way *arrived* and should be notified of the technical situation I am now in and what it means. And when a technically enabled "session" of some kind begins, the participants must be alerted to this *commencement* and given the opportunity to respond.

Given an understanding of the type of situation, a user may wish to modify her *status and availability*, and perhaps even her representation of *self-identity* accordingly. She may or may not wish to become a participant. She may or may not wish to make herself available to others in various ways. She may wish to convey only certain aspects of her identity, or remain anonymous or pseudonymous (Edwards, 1996). Thus she needs information about the *social rules* and their *technical implementations* that define what she is able to do within the current context-aware situation.

Our framework, by emphasizing these kinds of requirements, is intended to alert designers to the fact that it may sometimes be necessary to take a more holistic view of a context-aware infrastructure that goes beyond the component-based approach of Dey et al. With a component-based approach alone, which separates applications from their sensing and service logic, it is not possible to meet our requirements in building applications. Designers may also need to modify other components to ensure that they deliver the necessary information or offer the required behavior that is appropriate for the application in question.

### **Provide Feedback**

It is a basic principle of human–computer interaction that feedback should be built into applications. Yet, curiously, much of the innovative work to date on tangible (or physical) user interfaces and context-aware systems pays scant attention to this well-established requirement (Ishii & Ullmer, 1997; Newman et al., 1991; Want et al., 1999), leaving many difficult problems to be solved. For example, the PEPYS system, developed by Newman et al., recorded information and made (frequently incorrect) inferences about ongoing activities without giving users any feedback whatsoever within the context. One of the

authors of this essay, who took part in the evaluation study of this system, was surprised to learn that her movements had been tracked and recorded by this system prior to her volunteering to use the service it provided. Incidents like this caused some negative reactions among some of the researchers at EuroPARC, where this work took place.

The traditional graphical user interface (GUI) provides many resources, that we now take for granted, that serve the purpose of feedback including:

- Feedforward (e.g., flashing insertion points, cursors, pointers, handles, window highlighting, rubberbanding during resizing operations, and dialog boxes for confirmation of potentially irreversible actions).
- In-process feedback (e.g., progress bars and hourglass icons).
- Confirmation (e.g., displaying text, images or icons that have been moved or created in their location).

These resources all had to be explicitly designed to overcome difficulties that were commonly experienced with early user interfaces. They have now become so familiar, as we turn our attention away from the GUI as a design problem space, that it seems some researchers may be overlooking the problems they solve (Bellotti & Rodden, 2000). But those difficulties are likely to re-emerge in our experiences with novel tangible and context-aware systems unless we take explicit measures to avoid them.

In particular, users of context-aware systems need to know about, and be able to control, the following things that relate to them (Bellotti, 1997; Bellotti & Sellen, 1993):

- Capture: Whether and how has (or might) a context-aware system acquired information about me and my actions?
- Construction: What happens to my information once it is inside the system? Is it collated, cross-referenced, stored, or interpreted?
- Accessibility: To what or to whom is that information available?
- Purpose: To what use is this information put? In particular, is the information purely transient or is it being stored such that it might be used at a later date.

Conversely, users also need feedback about whether or not they are stumbling over elements in a context that relate to or belong to other people, and that might not necessarily be public, even if they are somehow accessible (cf. Cohen, 1994).

In many GUI-based collaborative systems that offer shared workspaces (e.g., Dourish & Bellotti, 1992; Edwards & Mynatt, 1997; Gutwin,

Greenberg, & Roseman, 1996; Neuwirth, Kaufer, Chandhok, & Morris, 1990), awareness cues are embodied in the workspace itself. These cues may reveal information about ownership of particular artifacts or indicate where colleagues are working in a shared space. In environments in which computation is not only pervasive, but may even be acting without our directed attention, we will certainly have even greater need for feedback about how our actions might affect others.

In particular, users need feedback about the availability of their own information to the system and to other users, and they must be in a position to respect the potential concerns of others in using the information the system provides them. Given these concerns, then, our framework calls for a designer to consider moving outside of concerns that are purely application-centric within Dey et al.'s framework. As a concrete example, in some contexts, it might be necessary to inform users as to the persistence of information retained by an aggregator. This is a system-constructed record of their activity and might potentially pose a risk to their privacy if the aggregator is communicating with other applications of which the user is unaware. And conversely, it might be important to make users aware that a system is automatically aggregating the information that they acquire about others, and to deter or prevent use of this information in the same manner as if it were being published explicitly by those others.

### **Enforce Identity and Action Disclosure**

Heath and Luff (1991) were among the first authors to highlight the problem of *disembodied conduct* in computer-mediated communication. This difficulty arises because computers allow actors to cause events to occur in a system with no automatic guarantee that the actor or the action can be detected. Bellotti and Sellen (1993) developed this argument to highlight that *dissociation* is an even more extreme phenomenon, whereby the identity of actors within the system may not be ascertainable at all. Dourish (2001 [this special issue]) also explores the problem of embodiment in some detail. Briefly here, the problem is that embodiment (or some equivalent virtual mechanisms) and socially responsible behavior are inextricably linked. First, because people need to perceive the actions and intentions of others in order to respect them, and, second, because they themselves need to be visible and thus accountable for their own actions to reduce the risk of antisocial behavior.

For these reasons, designers of context-aware systems must provide explicit mechanisms that enforce identity and action disclosure. Human-salient details that are necessary for this enforcement, then, will be *presence, identity, arrival, departure, status, availability, and activity*. Many of these details may be simple to build solely into an application component within Dey et al.'s architecture.

However, such an approach depends on the nature of the existing components within the architectural framework, and application designers must ensure that they perform the work necessary to enable these details to be reliably presented. Should it be the case that they do not, then designers must either modify the components as need be, or make it clear to the user where they fall short.

### **Provide User Control**

Earlier in this essay we argued that systems cannot be entrusted with the role of taking action on behalf of users, because there are too many contextually contingent considerations that shape human behavior. In other words, people's activities cannot be deterministically predicted. This could lead to two counterarguments:

- If systems don't do anything, there will be too many matters that users must deal with themselves, somewhat undermining the point of context-aware systems.
- Even if the system is enabled to take action, it will constantly be annoying the user with warnings or queries if it can't go ahead and do things on its own.

In fact there are already illuminating examples of "aware" systems that attempt to do things for people. As an example, most people are familiar with the experience of AutoCorrect and AutoFormat in the latest versions of Microsoft Word™. In many ways, Word has some awareness of what is being typed and attempts to "help" the user by taking action on their behalf using simple rules of spelling, grammar, and formatting. For most people that we authors know, this is often an infuriating experience. This problem is not purely because the system does things, but because it's so hard to stop it doing things or to correct them when it gets them wrong. In this case, the user does not have adequate control over the system.

Thus effective control is not simply about whether the user is intimately involved in execution (with constant user intervention and monitoring); it is more a matter of how easily the user attains the desired outcome (by whatever means). The degree to which user involvement is required, and the point in the interaction at which that involvement will be required, will depend on how much can be determined a priori about a particular action, its likely context, and its intent. Thus, to minimize the human effort required to attain desired outcomes, different design strategies for control are appropriate under different conditions:

- If there is only slight doubt about what the desired outcome might be, the user must be offered an effective means to *correct the system action*. In most cases the system will do the right thing (this is undoubtedly the case with the nonproblematic aspects of Microsoft Word's AutoCorrect and AutoFormat features).
- If there is significant doubt about the desired outcome, the user must be able to *confirm the action* the system intends to take.
- If there is no real basis for inferring the desired outcome, the user must be offered available *choices for system action*.

AutoCorrect and AutoFormat do not support these latter two strategies, leading to much frustration with repeated attempts to correct unwanted behavior.

From this familiar example it should be clear that systems that “do things” cannot be relied on to get it right and must always be able to defer to the user, particularly over questions of capture, construction, access, and purpose of information or in relation to mediating between people. This is all the more true when the results of system action are less obvious than an incorrectly formatted document, as is likely to be the case with context-aware systems.

We argue that application designers will often need to concern themselves with the details of sensors, context widgets, interpreters, aggregators, services, and their associated logic to ensure that the correct strategy for control is offered. Different components will have different weaknesses and failure modes, leading to different undesirable behavior that must be guarded against or effectively managed by applications and controlled by users.

Naturally, different communities will probably require different standards of user control over access to data or persons; thus the degree to which designers are required to compensate for shortcomings in system-mediated control will vary widely. For an interesting account of cultural variations of this sort see Dourish (1993).

#### 4. DISCUSSION AND CONCLUSIONS

This essay has presented a framework for designing context-aware systems that are both *intelligible* to their users and support the *accountability* of other users and the system itself. We believe that these are the crucial notions that must be present for context-aware systems to be useable, predictable, and safe. To support this framework, we have proposed a number of design principles (such as feedforward, disclosure, etc.) and several key aspects of context that must be available to support these principles.

From these principles, we can begin to envision a set of requirements for systems designed to support context-aware computing. For instance, although we subscribe to the software engineering value of separation of con-

cerns, such separation can prove damaging to the ability of a system to provide users with an account of its actions, interpretations, and decisions. This is particularly true in cases where inferences about user behavior or desire are separated from the applications with the domain knowledge necessary to situate such inferences.

Further, the very fact that some set of inputs can lead to multiple, plausible interpretations of a situation leads us to believe that complex systems for automatic interpretation are not appropriate. In fact, we would argue that systems that favor computation over representation—by using simplistic representations of contextual information, while relying on overly intelligent machine interpretations of that information—are destined to fail. A better approach is to support rich, fluid representations of context that capture the vagaries and the multilayered states of a situation *without* imposing interpretation on it. Such an arrangement, although perhaps less amenable to machine interpretation, leaves the user better informed to make decisions for him or herself, armed with full disclosure of the contextual information at hand.

---

## NOTES

**Acknowledgments.** We thank Rebecca Grinter, Paul Dourish, and Nicolas Ducheneaut for helpful comments and suggestions on drafts of this article.

**Authors' Present Addresses.** Victoria Bellotti, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. E-mail: [bellotti@parc.xerox.com](mailto:bellotti@parc.xerox.com). Keith Edwards, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304. E-mail: [kedwards@parc.xerox.com](mailto:kedwards@parc.xerox.com).

**HCI Editorial Record.** First manuscript received December 19, 2000. Accepted by Thomas Moran and Paul Dourish. Final manuscript received February 27, 2001. — *Editor*

---

## REFERENCES

- Abowd, G., Atkeson, C., Brotherton, J., Enqvist, T., Gully, P., & Lemon, J. (1998). Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. *Proceedings of CHI 98, Conference on Human Factors in Computing Systems*. New York: ACM.
- 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*, 5, 421–433.
- Agre, P., & Rotenberg, M. (1997). *Technology and privacy: The new landscape*. Cambridge, MA: MIT Press.

- Bellotti, V. (1997). Design for privacy in multimedia computing and communications environments. In P. Agre & M. Rotenberg (Eds.), *Technology and privacy: The new landscape* (pp. 62–98). Cambridge, MA: MIT Press.
- Bellotti, V., & Rodden. (2000). The perils of physical interaction. In W. Mackay (Ed.), *Proceedings of ACM DARE Designing Augmented Reality Environments*. Elsinore, Denmark.
- Bellotti, V., & Sellen, A. (1993). Design for privacy in ubiquitous computing environments. In G. de Michelis, C. Simone, & K. Schmidt (Eds.), *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW 93)*. Amsterdam: Kluwer.
- Bellotti, V., & Smith, I. (2000). Informing the design of an information management system with iterative fieldwork. *Proceedings of ACM DIS 2000 Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. New York: ACM.
- Boeing. (1995, June 14). *Boeing 777 digital design process earns technology award*. Available: <http://www.boeing.com/news/releases/1995/news.release.950614-a.html>
- Cohen, J. (1994). Out to lunch: Further adventures monitoring background activity. *Proceedings of ICAD 94, International Conference on Auditory Display*. Santa Fe, NM: Santa Fe Institute.
- Dey, A. K., Abowd, G. D., & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16, 97–166. [this special issue]
- Dey, A. K., Futakawa, M., Salber, D., & Abowd, G. D. (1999). The Conference Assistant: Combining context-awareness with wearable computing. *Proceedings of the Third International Symposium on Wearable Computers (ISWC 99)*. San Francisco: IEEE.
- Dourish, P. (1993). Culture and control in a media space. *Proceedings of the European Conference on Computer-Supported Cooperative Work, ECSCW 93*. Amsterdam: Kluwer.
- Dourish, P. (1997). Accounting for system behaviour: Representation, reflection and Resourceful action. In M. Kyng & L. Mathiassen (Eds.), *Computers and design in context* (pp. 145–170). Cambridge, MA: MIT Press.
- Dourish, P. (2001). Seeking a foundation for context-aware computing. *Human-Computer Interaction*, 16, 229–241.
- Dourish, P., & Bellotti, V. (1992). Awareness and coordination in shared workspaces. In *Proceedings of ACM CSCW 92 Conference on Computer Supported Cooperative Work*. Toronto, Canada. New York: ACM.
- Dunlop, C., & Kling, R. (1991). *Computerization and controversy: Value conflicts and social choices*. San Diego, CA: Academic.
- Edwards, W. K. (1994). Session management for collaborative applications. *Proceedings of ACM CSCW 94 Conference on Computer Supported Cooperative Work*. New York: ACM.
- Edwards, W. K. (1996). Policies and roles in collaborative applications. *Proceedings of ACM CSCW 96 Conference on Computer Supported Cooperative Work*. New York: ACM.
- Edwards, W. K., & Mynatt, E. D. (1997). Timewarp: Techniques for autonomous collaboration. *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*. Atlanta, GA.

- Goffman, E. (1959). *The presentation of self in everyday life*. New York: Doubleday.
- Goffman, E. (1963). *Behavior in public places: Notes on the social organization of gatherings*. New York: Macmillan.
- Gutwin, C., Greenberg, S., & Roseman, M. (1996). A usability study of awareness widgets in a shared workspace groupware system. *Proceedings of ACM CSCW 96 Computer-Supported Cooperative Work*. New York: ACM.
- Heath, C., & Luff, P. (1991). Disembodied conduct: Communication through video in a multimedia office environment. *Proceedings of ACM CHI 91 Conference on Human Factors in Computing Systems*. New York: ACM.
- Ishii, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*. New York: ACM.
- Neumann, P. G. (1995). *Computer related risks*. New York: Addison Wesley.
- Neuwirth, C. M., Kaufer, D. S., Chandhok, R., & Morris, J. H. (1990). Issues in the design of computer support for co-authoring and commenting. *Proceedings of the Conference on Computer Supported Cooperative Work*. New York: ACM.
- Newman, W., Eldridge, M., & Lamming, M. (1991). PEPYS: Generating autobiographies by automatic tracking. *Proceedings of ECSCW 91 European Conference on Computer Supported Cooperative Work*. Amsterdam: Kluwer.
- Rochelin, G. (1997). *Trapped in the net: The unanticipated consequences of computerization*. Princeton, NJ: Princeton University Press.
- Schilit B., Adams, N., & Want, R. (1994). Context-aware computing applications. *Proceedings of the First International Workshop on Mobile Computing Systems and Applications*. Santa Cruz, CA: IEEE.
- Suchman, L. (1987). *Plans and situated actions: The problem of human-machine communication*. New York: Cambridge University Press.
- Want, R., Fishkin, K., Gujar, A., & Harrison, B. (1999). Bridging physical and virtual worlds with electronic tags. *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*. Pittsburgh, PA: ACM.
- Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The Active Badge Location System. *ACM Transactions on Information Systems*, 10, 91-102.
- Wax, T. (1996). Red light, green light: Using peripheral awareness of availability to improve the timing of spontaneous communication. *Proceedings of ACM CSCW 96 Conference on Computer Supported Cooperative Work (Short papers)*. New York: ACM.

Copyright of Human-Computer Interaction is the property of Lawrence Erlbaum Associates and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.