People in our Software

JOHN RICHARDS AND JIM CHRISTENSEN,
IBM THOMAS J. WATSON RESEARCH CENTER

People are not well represented in today's software. With the exception of IM (instant messaging) clients, today's applications offer few clues that people are actually living beings. Static strings depict things associated with people like e-mail addresses, phone numbers, and home-page URLs. Applications also tend to show the same information about a person, no matter who is viewing it.

This information does not change, at least not very rapidly. If your only exposure to people were through these strings, you would have little reason to believe that people actually move about in physical and virtual space and *do* things.

What's wrong with this? First, dynamic information about people is often useful. If you are trying to meet up with someone, it is helpful to know that you are both in the same building rather than, say, 50 miles apart. Second, even static information about people is often sensitive. You may be willing to share your phone number with your professional colleagues, but prefer to keep it from those trying to sell you stuff.

Could people be made to "come alive" in software? If so, could access to this dynamic information be managed to respect personal privacy? And could we find compelling uses for this dynamic information?

The background is a dark, abstract composition. It features a faint, light-colored grid pattern that covers the entire area. Overlaid on this grid is a stylized, low-poly representation of a human face. The face is composed of various geometric shapes in shades of blue, green, and yellow. The eyes are particularly prominent, with one eye appearing more defined than the other. The overall effect is a blend of digital and organic elements, suggesting themes of artificial intelligence, data, and human interaction.

**A person-centric approach
could make software come
alive, but at what cost?**

People in our Software

For the last few years at IBM's Thomas J. Watson Research Center, we have been exploring the implications of embedding dynamic representations of people in our software. We began by investigating various rule-based schemes for routing personal messages based on what was "sensed" about a person's location and activity. Although people were not actually "depicted" as dynamic entities to end users in these early systems, their dynamic behavior modified the decisions made by the notification engine.

More recently, we worked to create a lightweight representation of people that can be directly embedded in a range of applications. Strongly influenced by the notion of social translucence,¹ we have explored how the cues provided by this easily embeddable representation can be directly used by other people (as opposed to being used by hidden computational apparatus). Before turning to the possible uses of these cues, we'll consider three problems with the notion of "live" people in our software: cost, privacy, and permission control.

COST

Making people come alive in our software requires the capture, aggregation, and publication of event streams generated by peoples' movements and activities. A number of different sensor technologies can detect peoples' locations. Detection becomes especially easy if people are willing to wear or carry a unique "badge" of some sort, typically based on RFID (radio frequency identification). The deployment of sensor arrays is not particularly cheap, however. So, without a compelling reason to know where people are, sensor-based location detection is not likely to catch on.

Activity detection is even harder than location detection. Sensor-based approaches to activity detection are neither cheap nor easy. Plus, there's the matter of what to look for as indicators of activity? Proximity to other people? To certain types of motion? To certain patterns of sound? Inferring higher-level activities from lower-level sensor data is problematic.

Fortunately, individuals are beginning to make both

location and activity detection cost effective by carrying network-connected devices with them. Pagers, cellphones, wireless PDAs, and laptops all make at least occasional network connections. In addition, the IP subnet accessed, wireless access point used, cell tower proximity, and other attributes of the network connection all provide clues to physical location without the cost of additional sensors. We not only carry these devices around with us, we also interact with them. Every touch is potentially an event worth reporting to the network. Simple hooks can be added, generally in the operating system itself, to determine which application is being used at any given moment. With these hooks, it is quite easy to determine that someone is doing e-mail, or Web surfing, or presenting a slide show, and so forth.

LOSS OF PRIVACY

Do you want such details of your life leaking into the network for others to access? Maybe yes, maybe no. If you are a member of a field service force, it's quite likely that you already report your location and activity to a central coordination point. It might be nice, however, to have the reporting done automatically. Or maybe it's fine for your spouse to know when you're leaving the office for home, but you would rather not share this information with people outside your family circle.

What we're comfortable revealing depends a lot on *to whom* we're revealing it. This context is both nuanced and highly dynamic. For example, you might find it helpful for people in your immediate work team to know when you are in the next room (with the possible exception of that one person who always complains every time you happen to be in sight). It might be useful to both you and the team if everyone knows you are presenting slides or writing code. Maybe this information would help them avoid interrupting you at a bad time. Maybe it could be used as a signal that a group meeting is starting, causing stragglers to leave their offices and gather in the meeting room.

Inclusion in a "trusted" circle could be based on many things. The mere fact that you are writing a joint paper,

for example, makes it permissible to reveal certain activities to one another. If you are approaching a deadline, it might be permissible for those contributing to the work to see more about each other than normal.

COMPLEXITY OF PERMISSION CONTROL

Just how much effort are you willing to put into controlling what can be seen by whom? A common approach to specifying “permissions” is through the writing of rules. Group membership, time of day, day of week, type of activity (perhaps as coded in a calendar), location, and so forth, are associated with specific permissions. For example, if it’s a workday, and if you’re a member of my workgroup, then you can see certain things about my location and activity. This makes a fair amount of sense. It is certainly appealing to think that the rules can be set up once and then just remain in the background causing the right decisions to be made in the future. Some rules are both easy to express and cause the appropriate information to be revealed in routine situations.

The problem is that the rules tend to become complex rather quickly. If it’s a workday, and if you’re a member of my workgroup, or if my calendar includes a meeting with you within one hour, then location can be seen—unless I am out on personal business. Complex rules can be specified, of course. GUIs (graphical user interfaces) can be constructed to make their specification reasonably tractable for many people. But even complex rules fail to adjust to the subtle nuances of actual contexts. Is it really all right for your workgroup to know that you’re presenting slides in the boss’s office? Again, it all depends. It’s probably fine if this is a regular progress update. But what if you are engaged in an escalation of a decision made by your team leader? Would you ever create a rule to capture this difference? Would you remember to tweak a more global rule prior to the meeting?

We believe there are ways to address these concerns. For example, two years of prototyping and trial deployments have shown that it is quite feasible to use existing portable devices as low-cost generators of useful location and activity information. We have also found that people can understand and control a privacy model based on simple person-to-person relationships. This will become more evident if you consider a particular (but illustrative and compelling) embodiment of these notions in which location and activity are used in the service of improved interpersonal communication.

THE GRAPEVINE MODEL

We have created a scalable, context aggregation and

publication infrastructure and a collection of Web services that we collectively call “Grapevine” (as in “I heard it on the...”). People can be embedded in applications through simple “person elements” (e.g., a custom person-tag in a Web application) that know how to connect to this infrastructure. One way that people can surface in applications that use this infrastructure is through an e-Card.

e-Card. An e-Card looks something like a business card, but in fact is an active window that displays up-to-date information about its “owner.” The card lists both “context” information (location and/or activity) and available “communication channels.” Figure 1 provides an example of Jim’s card, which states that Jim is currently near his office in the Hawthorne 1 building of IBM’s Watson Research Center and that he is currently involved with instant messaging. It also shows that four kinds of communication channels are available to John, the “viewer” of this card: telephone, IM, e-mail, and face-to-face dialog (by asking for time on Jim’s calendar).

Looking at Jim’s card, John might decide that now is a good time to reach Jim and that IM is a good means to use. On the other hand, knowing Jim well, John might decide to send an e-mail instead since Jim tends to have a dozen chats going at any one time. Alternatively, John may decide to walk up to Jim’s office if he’s also at IBM’s Hawthorne 1 building and has a sufficiently important topic to discuss with him.

Permissions for access to e-Cards can be based on relationship, group membership, or situation.

Permissions Based on Relationship. While e-Cards can make it easier for the “viewer” to communicate with the “owner” (by providing helpful context information and putting all communication channels only a single click away), the owner has a strong desire to retain control of both interruptions (in the form of attempts by others to communicate with them) and the location and activity information shown on the e-Card. Grapevine lets the

People in our Software

owner grant (or deny) permission for others to see and use the card. The examples below show how to adjust what a specific viewer (see figure 2) and a default viewer (see figure 3) can see and do with the e-Card.

Note that these permissions can be changed at any time, not just when the e-Card is first given to a viewer. Perhaps John got Jim's e-Card as part of an e-mail and decided to save it. Jim could decide later to turn off John's view of his location or to block phone calls or instant messages.

Note also that permissions are not necessarily symmetrical. In this particular case, Jim can have different permissions for John than John has for Jim. Although this may seem somewhat "unfair," it is in fact necessary as their relationship may not be symmetrical.

Finally, it should be noted that e-Cards both "federate" communication capabilities and "hide" the details of particular communication paths. Strictly speaking, this is not an aspect of the core privacy and permission model, but it does further the goal of protecting personal privacy. For example, if John decides to call Jim and clicks on the phone button, he never actually sees the phone number used to reach Jim. An automatic call-broker bridges John and Jim (based on its knowledge of Jim's location and telephony preferences) without revealing Jim's phone number. Should Jim decide to turn off John's phone access in the future, John can no longer call, as he

never knew Jim's actual number. This might be useful, for example, if John is a salesperson from whom Jim wants to receive one—and only one—call.

Permissions Based on Group. At times the manipulation of individual permissions is needlessly tedious. You may often want to manipulate permissions for entire groups. We note that the basic model used for per-person permissions need not be extended to accommodate these cases. All that is needed is a way to identify the set of people for whom the owner's per-person permissions should be adjusted. Examples of programs that define groups of people are IM and e-mail clients and address books. Once the group of people has been selected, their permissions could be adjusted using an interface (see figure 4).

Permissions Based on Situation. At times a person may wish to block all viewers' access to some communication channels or some location or activity information. In our current prototype, the e-Card's owner may request this sort of blocking interactively using the Grapevine context agent's window (see figure 5). It may also be done automatically. For instance, we allow e-Card owners to easily specify that no telephone calls or chats will be permitted when they are giving slide presentations. This has proven to be very useful (although the astute reader will note that we have slid a bit of rule-based specification back into the game). The screenshot of the Grapevine context agent window also demonstrates how telephone calls

from all e-Card viewers have been blocked with a single click.

The “Do Not Disturb” button at the bottom of the context agent window is an additional convenience for our users. A single click on this button causes all open e-Cards to change to the equivalent of a static business card with no location and activity information and no active communication channels. Individual e-Cards revert to their normal permissions when this button is clicked again.

THE GRAPEVINE AS A SYSTEM

At the system level, an e-Card is fundamentally just a pair of identities—owner and viewer—and a permission vector, which determines what context can be seen and which communication channels can be exercised.

Four attributes define the Grapevine system: complexity, context, interest, and speed of information.

Complexity. Not surprisingly, “identity” unpacks into a world of complexity. Just who is it that is asking the Grapevine infrastructure for a view of an e-Card? How is this identity established and authenticated? Must it be provided through a user name and password instead of being picked up automatically from some system context on the viewer’s device? If so, how often must it be reestablished in the course of a session? How reliably are various pseudonyms mapped to it? How resistant to spoofing is it?

Context. Context is also inherently rich. Interesting events, pushed from individual client devices, are consumed by individual e-Cards. But “interest” varies with device. For a wireless communicator it may be whenever the nearest radio tower changes. This might be detected by the device and pushed, or it might be pushed from an access point in the network itself. For laptop computers it may be whenever a network connection is made (with things such as subnet captured and pushed from the client) or whenever the wireless access point changes (with this being captured in the wireless network itself). In addition, laptops may want to push some aspects of the currently active application into the network, but we believe that the control over what is pushed and how it is represented on behalf of the owner of this data needs to be kept under the owner’s control.

Interest. This also varies with situation. Is it interesting to know that a person has just synched a handheld device? If this is part of the owner’s daily routine, then it might not be interesting at all. But it might be a very interesting event if it signals (for a particular viewer) the receipt of a critical document. Level of interest may also

hinge on correctly interpreting multiple-context streams. To take a simple example, you may have no idea what it means when someone is tunneled into the corporate intranet through a VPN (virtual private network), suggesting off-campus access. You may also not be able to know if a person has traveled to attend an out-of-town meeting scheduled on the calendar. But the two events in combination might make it clear that the travel did occur and the person is currently in or near a particular city. At this time, we do not know how to deal with these sorts of complexities. We just expose (again, subject to an individual’s permission) a small set of relatively low-level events associated with events generated by people using particular applications on particular devices. We must rely on the viewer’s understanding of the larger shared world to interpret these events in a meaningful way. Further research may explore how to rank events by interest or at least how to progressively disclose events not directly exposed in an e-Card by default.

Speed of Information. An additional system challenge involves the rapid and efficient movement of

People in our Software

large amounts of permission-filtered (hence, not easily broadcast) context information through the network. We currently achieve the distribution of filtered location and activity information and the updating of communication channel availability through a JMS (Java Messaging Service) infrastructure. For scalability, the awareness servers publish a single version of a person's dynamic state and let the messaging infrastructure filter it so that only those subscribers with sufficient privileges receive it. JMS provides both access control lists and message selectors (using a subset of the SQL-92 syntax) to achieve selective delivery of messages to authorized clients. In addition, JMS topics provide an abstraction for the rendezvous between the awareness server hosting any one registered Grapevine user, and all sources and viewers of awareness information for that registered user. Whereas early Grapevine prototypes used only SOAP (Simple Object Access Protocol) to interface to the awareness servers, our current system uses JMS to get information to and from the servers in both "push-only" and RPC (remote procedure call) styles of access. We continue to use SOAP to access Web-based sub-services and for rapid prototyping of new applications containing "live people."

A FUTURE OF IMPROVED COMMUNICATION

Grapevine e-Cards are one embodiment of the more general notion of live people in software. Going beyond mere "presence" (which tends to be confined to activity within a single application), Grapevine shows how multiple, inexpensive sources of location and activity information can be aggregated in the network and published with acceptable and understandable privacy controls. Grapevine e-Cards, in particular, demonstrate how this enhanced awareness can lead to improved interpersonal communication.

The availability of dynamic behavioral cues in software affords the possibility of making more informed inferences about what people are actually doing. Interactions with them can be better coordinated and a sense of shared community can be fostered. With careful design, it

is possible to achieve a sense of intimacy without becoming intrusive and threatening. If people find the notion valuable, and the cost of adapting existing applications is low, we believe the incorporation of live people in software will gain widespread acceptance. Q

REFERENCES

1. Erickson, T., and Kellogg, W. A. Social translucence: An approach to designing systems that mesh with social processes. *Transactions on Computer-Human Interaction* 7, 1 (March 2000), 59-83.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

JOHN RICHARDS joined the computer science research staff at IBM's Thomas J. Watson Research Center in 1978 after receiving his Ph.D. in cognitive psychology. He has served in research, design, and management roles in numerous application and interpersonal communications projects and has been recognized for his contributions in the area of digital voice-mail systems by the Human Factors Society. He is active in both ACM SIGCHI (Special Interest Group on Computer-Human Interaction) and ACM SIGPLAN (Special Interest Group on Programming Languages). He chaired the OOPSLA'91 conference and served as chair of the OOPSLA (Object-oriented Programming, Systems, Languages, and Applications) steering committee from 1991 to 1996. Richards was elected a Fellow of the ACM in 1997.

JIM CHRISTENSEN joined IBM in 1978 after receiving his M.S. in computer science from the University of Illinois at Champaign-Urbana. He joined IBM's Thomas J. Watson Research Center in 1983, where he has worked on a variety of projects in programming environments, understanding program execution, digital imaging and federated multimedia libraries, and context-aware applications that help people communicate. Christensen has held both management and engineering positions in his career with IBM and has received numerous awards.

© 2004 ACM 1542-7730/04/0200 \$5.00