Toward Context-Aware Computing: Experiences and Lessons

Joshua Anhalt, Asim Smailagic, Daniel P. Siewiorek, and Francine Gemperle, *Carnegie Mellon University*
Daniel Salber, Sam Weber, Jim Beck, and Jim Jennings, *IBM T.J. Watson Research Center*

To minimize distractions, a pervasive-computing environment must be context aware. The authors define an activity–attention framework for context-aware computing, discuss the spatial and temporal aspects of applications they developed, and introduce a pervasive-computing architecture.

The effects of Moore's law are apparent everywhere: Chip density, processor speed, memory cost, disk capacity, and network bandwidth are improving relentlessly. As computing costs plummet, a resource that we have ignored until now becomes the limiting factor in computer systems—user attention, namely a person's ability to focus on his or her primary task.

Distractions occur especially in mobile environments, because walking, driving, or other real-world interactions often preoccupy the user. A pervasive-computing environment that minimizes distraction must be context aware, and a pervasive-computing system must know the user's state to accommodate his or her needs.

Context-aware applications provide at least two fundamental services: spatial awareness and temporal awareness. Spatially aware applications consider a user's relative and absolute position and orientation. Temporally aware applications consider the time schedules of public and private events. With an interdisciplinary class of Carnegie Mellon University (CMU) students, we developed and implemented a context-aware, pervasive-computing environment that minimizes distraction and facilitates collaborative design.

Our approach

To identify the types of distraction that occur during the design process, we created an activity–attention matrix—the Distraction Matrix (see Figure 1). The Distraction Matrix categorizes activities as information (active and passive), communication (artificial, formal, and informal), and creation (contribution). Subcategories specify the types of primary activity within each category. For example, receiving information is a type of active-information activity, and initiating communication is a type of artificial-communication activity.

We based each distraction's location on how long it interrupts a primary activity. We categorized interruption durations as *snap*, *pause*, *tangent*, and *extended*. A snap distraction is one you usually complete in a few seconds, such as checking your watch; it should not interrupt your primary activity. A pause distraction involves stopping the primary activity, switching to a related one, and then switching back within a few minutes. Pulling over to the side of the road and checking directions is an example. A tangent distraction, such as receiving an unrelated phone call, is of medium duration and is unrelated to your primary activity. An extended distraction, such as stopping at a motel and resting for the night, is a relatively long-term interruption of your primary activity.

Applications

We equipped the campus with 400 wireless-networking access points, enabling wireless coverage for the entire campus. To move distractions toward the Distraction Matrix's left (snap) side, we implemented a complementary set of interactive applications and services that support mobile team-design activities. (See the related sidebar for information on relevant work in context-aware computing.)

Portable Help Desk. Because they have many meetings at various times and locations, students are often

1094-7167/01/$10.00 © 2001 IEEE

Time

		Snap	Pause	Tangent	Extended
Information					
Active	Receiving	Message arrival			Audio, Walkman
	Notifying	Information access			Transferring files from network
	Monitoring	Auction			
	Serendipity	Stocks, sports, matching similar needs			Reading news
		Free food			
Passive	Seeking	Line length	Exam calendar	Looking for class notes	
		Bus arrival	Software or hardware help	Who else is doing this now?	
		Locate person	Calendaring	Access personal data	
			Navigation		
	Browsing		Information on Web or built environment	Poster, bulletin board information	Web research
	Finding				Reviewing class notes
	Verifying		Recall previous queries		
			Double-checking information		
Communication					
Artificial	Initiating	S.O.S. emergency	Introductions	Team building	Chatting (public or private)
				Collaborative work	
	Participating	Instant messaging	Queries	Event planning	
				Assassins game	
				Social planning	
Informal	Broadcasting		Information exchange	Posting information to bulletin board	
			Scheduling		
				Advertising	
Formal		One-to-one communication	One-to-one communication	One-to-one communication	One-to-one communication
		One-to-group communication	One-to-group communication	One-to-group communication	One-to-group communication
		One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people
Creation					
Contribution	Recording	Remember this!		Class note taking	Generating messages
		Add a to-do or call list		Meeting	
	Synthesizing		Forwarding x to y	Filling out survey	Summarizing lecture
				Registration	
	Generating			New ideas	Mobile-tool building
				Adding information to existing projects	

Figure. 1. The Distraction Matrix. We based each distraction's location on the primary activity it interrupts and that interruption's duration (increasing from left to right).

unsure of where their next meeting is supposed to take place. The ability to observe team members' locations on campus helps students determine a meeting's location. The Portable Help Desk (PHD) application, a spatially aware system, confers that ability. It lets a user build maps of the immediate area, including colleague and static- and dynamic-resource locations, and quickly retrieve contact and resource availability information. While tracking a user's colleague, PHD displays that colleague's contact information.

The application can also display printer queues, restaurant hours, and stock of carbonated beverages and food in connected vending machines. Figure 2 shows the activities the PHD system supports and the attention each demands of the user.

We built both visual and audio interfaces for the PHD, each of which supports users in different contexts. The visual interface, designed for stationary use, is richer, but for a user who is walking around, the hands-free audio interface, Speech-PHD, is less distracting.

Figure 3 illustrates PHD's visual interface. The user selects people and resources in the left pane, and information about those people and resources appears in the middle pane. The right pane displays a campus map locating the selected people and resources.

Speech-PHD accesses the same database as the visual interface, so all responses are formatted similarly. Figure 4 is a transcript of the same queries that Figure 3 demonstrates. Because PHD knows the user's current location, it can answer questions such as "Where is the nearest ATM?"

PHD delivers information to the user in both proactive and user-driven manners. A user receives proactive information when engaging infrastructure resources such as printers. For instance, when the user begins a print job, PHD will alert him or her if a large print queue exists and suggest a nearby printer with a shorter queue. PHD can also suggest a printer near a destination to which a user is en route.

In terms of user-driven information, a design group waiting for a colleague can use PHD to locate the missing colleague and estimate his or her arrival time. The group also has access to the colleague's phone numbers. Essentially, PHD helps a group avoid repeating the beginning of a meeting for every late member. When the team members are getting hungry, they can look up the hours of nearby restaurants or check whether the soda machine is full.

Matchmaker. For large projects and design groups, no single individual has the expertise to perform every task. The Matchmaker application lets a user rapidly identify an expert user with the knowledge to help solve a problem. An expert's suitability depends on many factors, such as technical expertise, friendliness, proximity, and availability. Matchmaker infers expertise and skills by observing an expert's track record rather than by asking him or her explicitly. Matchmaker uses temporal context to determine an expert's availability and spatial context to determine the expert's distance from the user.

The Matchmaker system connects a user's query with an expert user who

- is nearby,
- is available,
- has a profile listing the skills needed, and
- has a history of answering similar questions.

Because the located expert is near the user initiating the question, he or she avoids wasting time moving to the user. After contacting the expert with the question, the Matchmaker system requests feedback from the expert to determine if he or she is best suited to answer the question. The database then updates its profile of the queried expert to increase expert-selection accuracy.

We have instantiated the Matchmaker system, letting users efficiently contact CMU's School of Computer Science Computing Support Group to resolve queries. The CSG maintains an extensive database of previously answered queries; this information lets the Matchmaker system generate profiles of CSG experts. Figure 5 shows some of the activities Matchmaker supports.

Figure 6 shows Matchmaker's system architecture. Matchmaker sends the user's query and the problem's location to the server. The server sends the query to the information-retrieval partition, which searches the database for similar queries, experts who answered those queries, and experts with similar knowledge. The central server sends the returned list of experts to the matchmaking partition, which compares the experts' locations and schedules

The Roots of Context-Aware Computing

Steve Mann introduced *humanistic intelligence*,[1,2] proposing it as a new signal-processing framework in which the processing apparatus supports and depends on the user's natural capabilities of body and mind. (For more on humanistic intelligence, see the Guest Editor's Introduction in this issue.) Anind K. Dey, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa designed a software architecture to let developers create context-aware applications.[3,4] Thad Starner, Bernt Schiele, and Alex Pentland developed context-aware user interfaces that use body-mounted, environment-looking cameras and machine-vision techniques.[5] Kristof Van Laerhoven and Ozan Cakmakci use body-mounted sensors to determine a user's activity and infer the user's context.[6] Gerd Kortuem, Zary Segall, and Martin Bauer describe a wearable computer that alters its user interface based on devices and services in the user's environment.[7]

References

1. S. Mann, "Humanistic Intelligence: 'WearComp' as a New Framework and Application for Intelligent Signal Processing," *Proc. IEEE*, vol. 86, no. 11, Nov. 1998, pp. 2123–2151.
2. S. Mann, "Humanistic Intelligence," *From the 1997 Ars Electronica Catalog*, www.wearcam.org/ars/hi.html (current 7 June 2001).
3. A.K. Dey, D. Salber, and G.D. Abowd, "Context-based Infrastructure for Smart Environments," *Proc. 1st Int'l Workshop on Managing Interactions in Smart Environments* (MANSE 99), Springer-Verlag, New York, 1999, pp. 114–128.
4. A.K. Dey et al., "An Architecture to Support Context-Aware Applications," tech. report GIT-GVU-99-23, Graphics, Visualization, and Usability Center, Georgia Tech, Atlanta, 1999.
5. T. Starner, B. Schiele, and A. Pentland, "Visual Contextual Awareness in Wearable Computing," *Proc. 2nd Int'l Symp. Wearable Computers* (ISWC 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 50–57.
6. K. Van Laerhoven and O. Cakmakci, "What Shall We Teach Our Pants?," *Proc. 4th Int'l Symp. Wearable Computers* (ISWC 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp.77–83.
7. G. Kortuem, Z. Segall, and M. Bauer, "Context-Aware, Adaptive Wearable Computers as Remote Interfaces to 'Intelligent' Environments," *Proc. 2nd Int'l Symp. Wearable Computers* (ISWC 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 58–65.

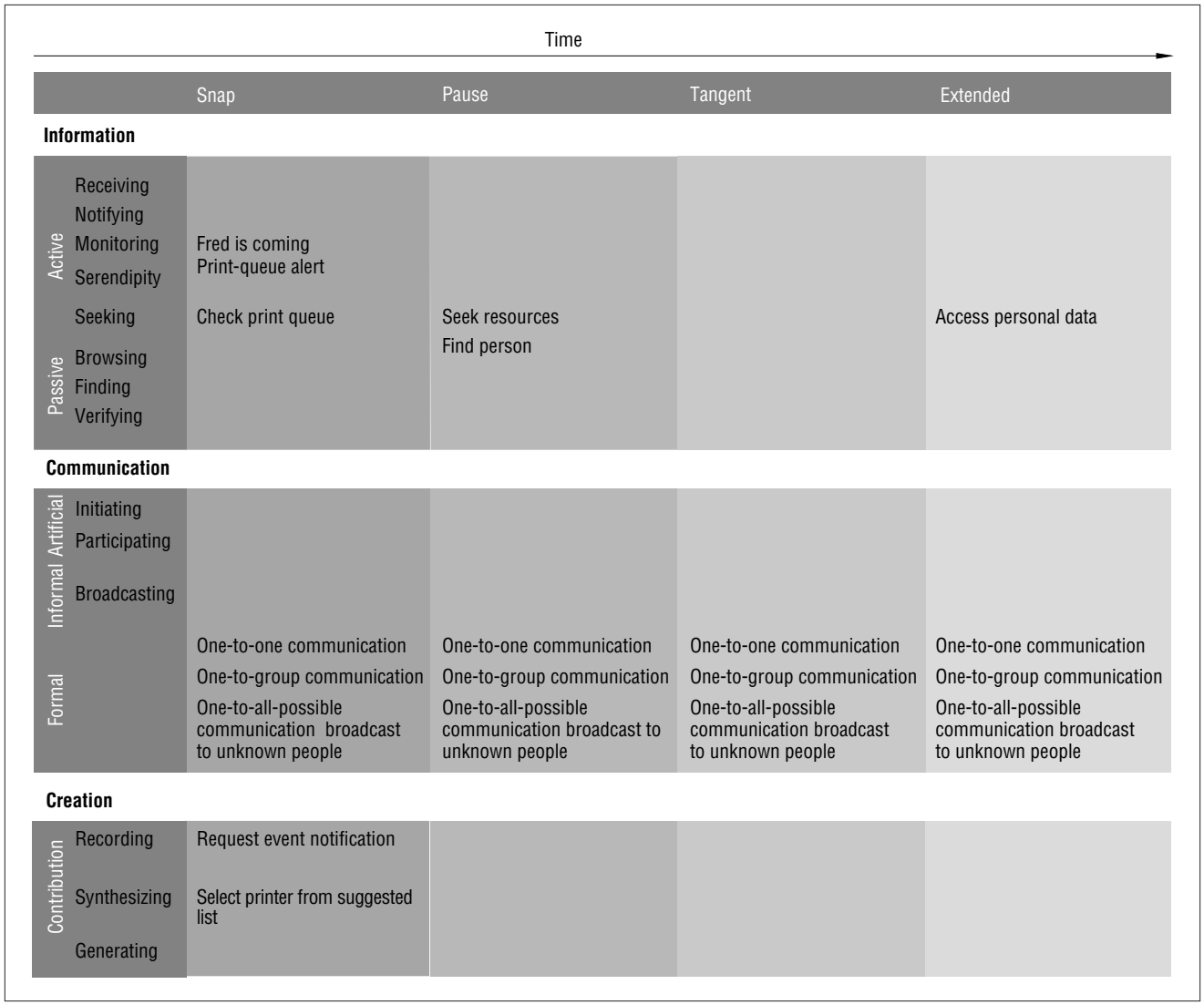

Figure 2. Distraction Matrix for Portable Help Desk.

to the query's context. Then, the system notifies the chosen expert of the query.

Privacy Guard. PHD offers a valuable service to collaborating work groups, but its location-sensing ability is a liability. Therefore, we developed Privacy Guard to help users protect their information. Privacy Guard enables basic privacy policies and advanced expressions describing which users, groups, and time periods PHD can and cannot report. Figure 7 illustrates Privacy Guard's architecture.

The location-sensing service client derives the user's location from the wireless-network card and sends that location to the central server. Users update the central server with permissions. When the server receives a query for a user's location, it compares the client against the target's permissions. Accordingly, the server then sends the client the target's location or a refusal to answer the request.

Context-aware agents. Busy groups tend not to have abundant time to browse calendars, check for new email, or read bulletin boards. Therefore, we developed context-aware agents to deliver relevant information when a user needs that information. When the user is not engaged in more important activities, context-aware agents display appointments, urgent emails, and interesting calendar events. The agents are proactive: they monitor public and private calendars and email accounts and deliver information to the user instead of requiring the user to poll the relevant sources. Their goal is to provide intelligent calendar management, including setting schedules and resolving conflicts with other users' calendars while accounting for the location and available resources for a meeting. We have implemented the following three agents:

- *Notification Agent* alerts a user who passes within a certain distance of a location that a task on his or her to-do list identifies. For example, if a user is near his or her mailbox, the agent alerts the user if a package is waiting.
- *Meeting-Reminder Agent* alerts a user who is likely to miss a meeting. The system identifies the time the meeting will start and determines the travel time there from the user's current location.
- *Activity-Recommendation Agent* recommends activities and meetings, based on the user's interests, that the user might like to attend. For example, consider a user who sets his Activity-Recommendation Agent to inform him when free food is available. As the user walks through a building, the system identifies a meeting with free food upstairs and notifies the user.

Figure 8 shows an example user interface for the Activity-Recommendation Agent.

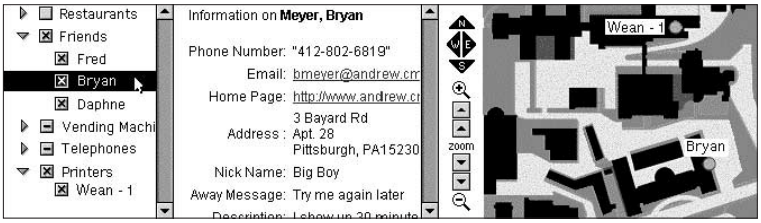

Figure 3. Portable Help Desk visual interface.

User:	"Locate Bryan."
Speech-PHD:	"Bryan is located in Hamburg Hall."
User:	"What is Bryan's phone number?"
Speech-PHD:	"Bryan's phone number is 412-802-6819."

Figure 4. Transcript from Portable Help Desk audio interface Speech-PHD.

The user defines his or her interests, enabling the agent to recommend upcoming activities. The agent categorizes interests by activity and keywords. The user can access the interface to find upcoming recommendations if he or she doesn't want to wait for the Activity-Recommendation Agent's notification.

We designed the context-aware agents to function as services that simpler applications can use.

The pervasive-computing environment

Mobile computing poses challenges such as intermittent and variable-bandwidth connectivity and client-resource constraints imposed by weight and size considerations. We based our pervasive-software architecture on one that the IBM T.J. Watson Research Center proposed. The class's prototype used Hewlett-Packard Jornada 680 palmtop computers and Itsy/Cue wearable computers communicating through Lucent Wavelan cards on Wireless Andrew.[1]

Our architecture's main goal is to let users seamlessly move work between devices. The architecture moves the application to a network-connected server, leaving only a minimal interface on the client device. Any device implementing the interface can then reattach the server running

Time

		Snap	Pause	Tangent	Extended
Information					
Active	Receiving				
	Notifying	Receive request for help			
	Monitoring	Completion of task notification			
	Serendipity				
	Seeking		User searches through list of possible solutions returned by system	User tries suggested solutions	
			Finding someone to help		
Passive	Browsing				
	Finding				
	Verifying				
Communication					
Informal	Artificial: Initiating				Expert and user collaborate
	Artificial: Participating				
	Broadcasting				
Formal		One-to-one communication	One-to-one communication	One-to-one communication	One-to-one communication
		One-to-group communication	One-to-group communication	One-to-group communication	One-to-group communication
		One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people	One-to-all-possible communication broadcast to unknown people
Creation					
Contribution	Recording		User initiates query		
	Synthesizing				
	Generating				

Figure 5. Distraction Matrix for Matchmaker.

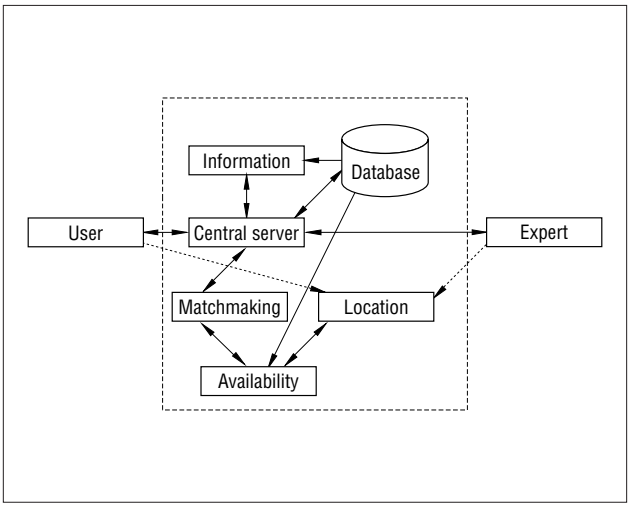

Figure 6. Matchmaker system architecture. The system receives the user's query and matches it to an appropriate expert user. It then locates the expert and notifies him or her of the query.

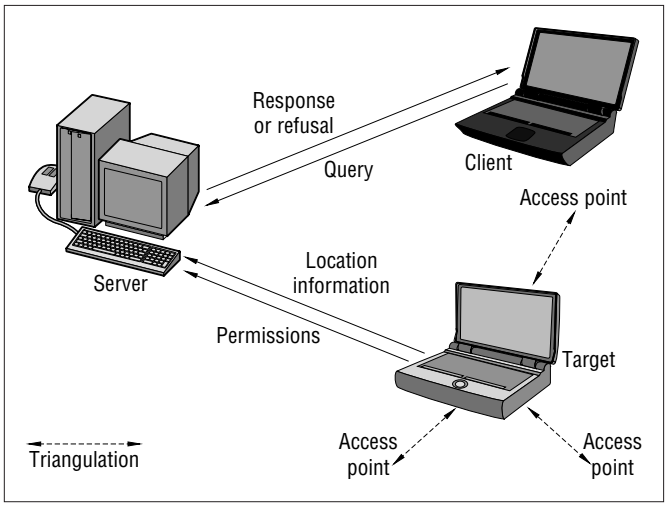

Figure 7. Privacy Guard architecture. By incorporating permissions, Privacy Guard limits a client user's access to a target user's contact and location information.

the application. To address mobile computing's connectivity issues, the architecture includes elements that preserve dataflow between the server and the devices. Optimizing data for device capabilities maximizes performance.

Original architecture

Figure 9 shows the four layers of IBM's original architecture. The bottom layer includes a range of mobile and fixed devices; neither hardware architecture nor operating system must be homogeneous. The second layer contains device proxies, which every device has and which represent a transcoding layer for each device. The third layer is the user-proxy layer. Every user has a personal user proxy. This layer can store applications and a user's state. The fourth layer is the services layer, where the architecture implements shared applications, utilities, and servers. All requests between layers are in hypertext transfer protocol (HTTP). The requests can include data structures such as integers, characters, and strings. Each request includes user and device identification.

IBM implemented the architecture in Java. A device executes the service manager, which prepares requests and interprets responses for client applications running on the device. The device proxies use WEBI, an HTTP proxy that IBM developed. This proxy intercepts a user's requests, passes them through a series of user-specified filters, and forwards the transcoded requests and responses. The user proxy receives a request and either starts an appli-

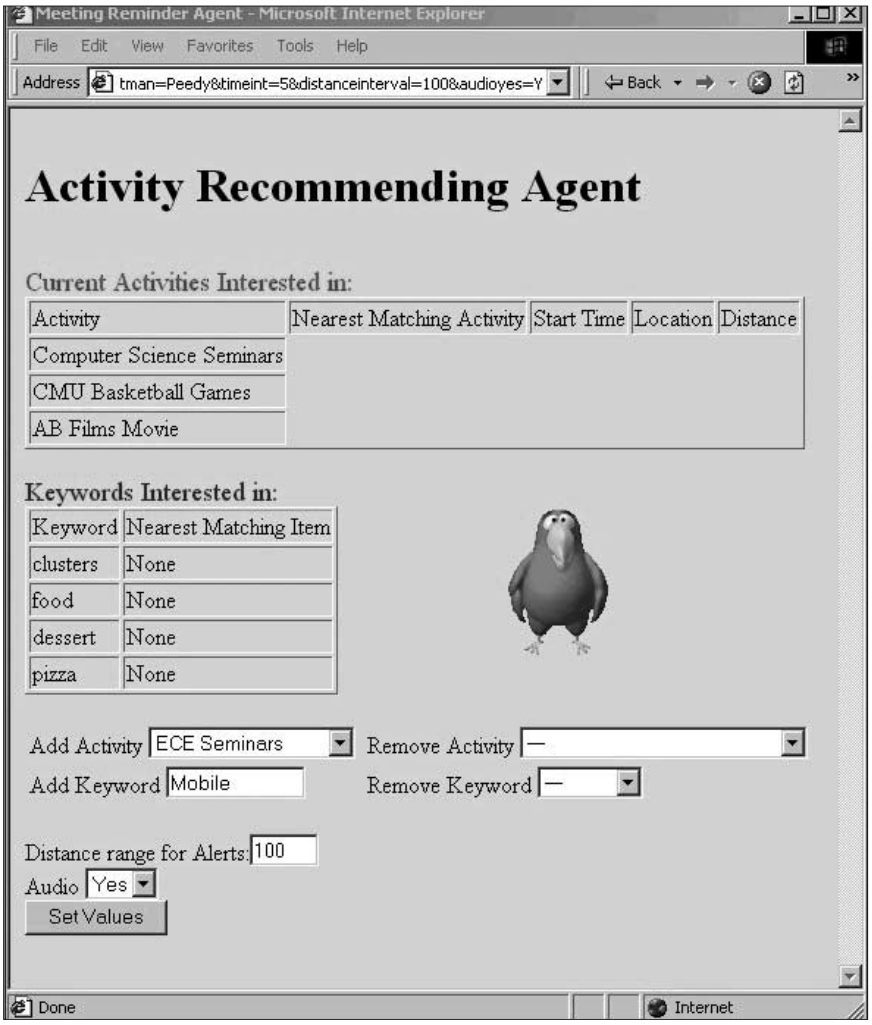

Figure 8. An example of an Activity-Recommendation Agent's user interface.

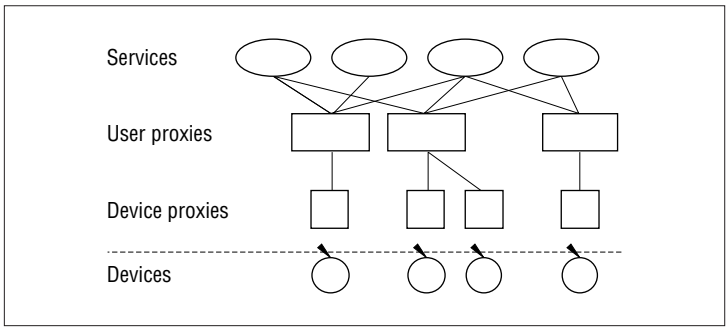

Figure 9. IBM's original architecture. Information passes from devices to services and back through device and user proxies.

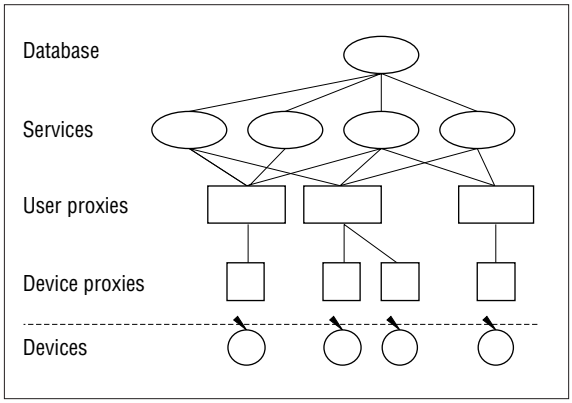

Figure 10. Handy Andy architecture. We added a fifth layer, a unified database that acts as a service, to IBM's original architecture (see Figure 9).

cation or forwards the request to a service. If the requested service is not well-known, or if the user's preferences don't define it, the user proxy invokes the Service Location Protocol to locate the requested service.

Revised architecture

The IBM architecture lets a service or user proxy store preferences and long-term state. We made the decision to create a unified (SQL) database as a service. We added a fifth layer to the architecture, above the services (see Figure 10). We named the new architecture Handy Andy, after Handheld Andrew, CMU's wireless-network project. In the Handy Andy architecture, all services' and user proxies' database access is based on the privileges of the user authenticated to them. That prevents common data such as a user's name, address, and contact information from being duplicated across systems. Users can update their data with a single application.

With the original architecture, conflicts pertaining to the format of stored information became apparent, as competing applications preferred certain data sizes and types. The revised architecture lets stored procedures translate stored data into any format that a service requests.

Idealink is a virtual meeting space tool. The user interface—a shared whiteboard that a user can archive for later review—is optimized for the minimal screen area that portable devices provide. The system operates within a client–server architecture: The application runs on the target devices, to which the server distributes screen updates. The Handy Andy architecture enables additional features and ease of implementation within a pervasive, wireless environment. Figure 11 shows a typical Idealink session's architecture elements within the Handy Andy architecture.

The Handy Andy architecture lets the system be more flexible. Also, the architecture automatically deals with problems inherent in wireless networks. Each user has one or more devices running the Idealink user interface. The devices might have color or black-and-white displays; their screen sizes might range from a watch-sized liquid crystal display to a wall-sized projected image. The architecture's device proxy adjusts color depth according to the device properties and instantiates filters that scale the size of screen updates. The devices do not use valuable clock cycles and battery power for these operations. If the communications channel between the device proxy and the device is broken, the device proxy caches updates until the device reestablishes connection. If the user so desires, he or she can start the Idealink session on one device and continue it on any other. From the user's calendar, the user proxy knows what meeting is taking place—which lets the system automatically negotiate who is included in the Idealink session. The user proxy stores preferences, including tool palette layout, and user-selected keystroke combinations. The Idealink service combines each user's additions to the session and updates each client. At the end of the meeting, the service archives the session in the database.

Location-sensing service

The Location Service generates a key parameter of context information. To determine a user's location, the wireless-network card (acting as a sensor) in the user's computer measures the signal strengths to all available wireless-access points and compares them to recorded training signal strengths. For every location, the sensor records a unique signal-strength reading from a group of access points. For training, the user manually inputs his or her location into the computer. The computer then takes and averages approximately 17 samples. This process generates a table that lists what signal levels to expect at different locations. The sensor requires only a single test, which it can save for use in later sessions and on other platforms. During use, the computer compares measured values to those in the table and computes differences. It reads the entry with the smallest difference as the current position.[2] As with Privacy Guard, the user requesting a target's location sends his or her request to a server. The server might use a caching mechanism to answer the request, or it might send the request to the target user. The target user's computer determines its location and sends the results to the server. The server completes the transaction by sending the target's location to the requesting user (assuming that user has permission to receive the target's information). Our Location Service is significantly more accurate than standard Global Positioning Systems.[2]

Table 1 presents the accuracy of our location-measurement results. We inferred accuracy from the fact that the distance needed for a signal-strength change of one decibel milliwatt (dBm) has been empirically determined to be approximately five feet when near an access point. Because more than 99.9 percent of our measurements are within 3 dBm of the actual value, we infer that the reported locations are within +/- 15 feet of actual positions.

Client–server speech issues

Speech-PHD requires significant computing resources for the automatic speech recognizer (CMU's Sphinx ASR—see www.speech.cs.cmu.edu for details) and for text-to-speech conversion (Festival Text-to-Speech software). When we were developing Speech-

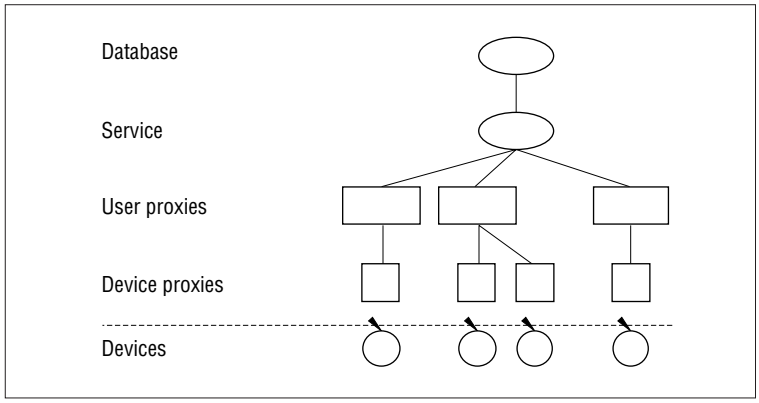

Figure 11. Idealink architecture within the Handy Andy architecture. The Handy Andy architecture provides a pervasive, wireless environment for the Idealink whiteboard tool.

PHD, no mobile device had the required computing power, memory, and nonvolatile storage. Placing the ASR and text-to-speech software on a server solved the resource limitations but introduced network latency. We measured latency from the end of the user's query until the system began its response. Table 2 summarizes the results. Transferring both the query and the response as a file required almost five seconds. By modifying Sphinx to stream the query, we reduced latency to two seconds. By modifying Festival to also stream the response, we reduced the delay by a factor of 25.

Lessons learned

The Handy Andy architecture provides a useful framework for developing persistent applications. The architecture is extremely broad in its description, letting developers implement very portable applications. A successfully implemented device proxy can maximize a device's usefulness while offloading expensive conversions to a network server. Developers can implement simple applications in the user proxy or as a service.

Developers have tried to make device proxies do more than they could. To fully exploit the capabilities of device proxies within the architecture, we explored speech- and user-interface-adaptation filters. ASR vocabularies are limited. Most require knowledge of the user's language in the form of a language model. The application must provide such a model; therefore, accessing the device proxies would no longer be transparent to the application programmer.

Our database lets users and services share data. Because we could customize the data-access interface for each service, we limited issues with proprietary application-protocol interfaces. The data inherited the database's security model, allowing user-permission specification and enforcement. This was convenient for programming, but it introduced a degree of failure: the database limited performance and exposed all shared data to security risks.

Table 1. Accuracy of location measurements.

Accuracy (%)	Strength (dBm)	Distance (feet)
68.6	+/– 0.939	+/– 5
95.4	+/– 1.146	+/– 10
99.9	+/– 2.817	+/– 15

Table 2. Speech-PHD network speech latency.

Latency (sec)	Transfer file query	Streaming query
Transfer file response	5	2
Streaming response	–	.2

Our proactive agents cannot access system-level functions, such as starting new applications, on the user's behalf. We won't let them do so until we address system-level security. Although context information helps generate more-intelligent system behavior, it is a liability for the system's users. Location information provides a prime example. All system levels, including the architecture, protocols, inferred preferences, and user-specified preferences, must address the security of such information.

We have not yet optimized the location service. Requiring the tracked client to return its current location for every request uses mobile devices' limited power and computation cycles. Ideas for increasing the location service's efficiency and scalability include caching and predicting user location. ■

Acknowledgments

We acknowledge the funding support of IBM Research, the US National Science Foundation under grant 9901321, and the Defense Advanced Research Projects Agency. We also acknowledge the students from the Rapid Prototyping of Computer Systems and Mobile Computing courses (Spring and Fall semesters) for their contributions to the project. We thank Mike Karasick for his continuous support to the project.

References

1. A. Smailagic and D. Siewiorek, "User-Centered Interdisciplinary Design of Wearable Computers," *ACM Mobile Computing and Comm. Rev.*, vol. 3, no. 3, July 1999, pp. 43–52.

2. J. Small, A. Smailagic, and D.P. Siewiorek, "Determining User Location for Context-Aware Computing through the Use of a Wireless LAN Infrastructure," submitted for publication to *ACM Mobile Networks and Applications*, vol. 6, 2001.

The Authors

Joshua Anhalt is a graduate student of electrical and computer engineering at Carnegie Mellon University. His research interests include ubiquitous computing, context-aware computing, and the privacy of information. He received his BS in electrical and computer engineering from Carnegie Mellon University. Contact him at 5260 Forbes Ave., Pittsburgh, PA 15217-1102; anhalt@andrew.cmu.edu.

Asim Smailagic is a faculty member at Carnegie Mellon University's Institute for Complex Engineered Systems, College of Engineering, and Department of Electrical and Computer Engineering. He also directs CMU's lab for Interactive Computer Systems. His research interests include pervasive computing, system-level design of advanced computer systems, rapid prototyping of wearable computers, and audio-visual and sensor interfaces to computers. Contact him at Institute for Complex Engineered Systems, Carnegie Mellon Univ., Hamburg Hall, Pittsburgh, PA 15213; asim@cs.cmu.edu.

Daniel P. Siewiorek, Buhl University Professor of Computer Science and Electrical and Computer Engineering at Carnegie Mellon University, directs the Human–Computer Interaction Institute. His research interests are mobile computing, computer architecture, human–computer interactions, and reliability. He received his BS in electrical engineering from the University of Michigan and his MS and PhD in electrical engineering from Stanford. He is a member of the ACM, Tau Beta Pi, Eta Kappa Nu, Sigma Xi, and the IEEE Computer Society. Contact him at Human–Computer Interaction Inst., CMU, Newell-Simon Hall, Pittsburgh, PA 15213; dps@cs.cmu.edu.

Francine Gemperle is a senior industrial designer with the Wearable Computing Group at Carnegie Mellon University. She is an expert in designing products for wearability and is working on becoming an expert in tactile interaction design. She received her BFA in industrial design from Carnegie Mellon University. She is a member of the Industrial Designers Society of America. Contact her at Carnegie Mellon University, Institute for Complex Engineered Systems, 2203 Hamburg Hall, Pittsburgh, PA 15213.

Daniel Salber is a research associate at the IBM T.J. Watson Research Center. He is interested in software engineering for human–computer interaction and context-aware computing. He received his MS and PhD in computer science from the University of Grenoble. He is a member of the IFIP Working Group 2.7 on Engineering for Human–Computer Interaction, ACM SIGCHI, and a founding member of AFIHM. Contact him at IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532; salber@acm.org.

Sam Weber is a research staff member at IBM's T.J. Watson Research Center. His research interests include software architecture, security, and verification. He received his BSc and MSc from the University of Toronto and MSc and PhD from Cornell University. His hobbies include swing and contra dancing and collecting mechanical puzzles. Contact him at IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532; samweber@watson.ibm.com.

Jim Beck is director of systems engineering and mobility at Inmedius Corp. and a visiting researcher at CMU's Human–Computer Interaction Inst. His research is in mobile computing, *n*-tier distributed applications, service frameworks, component-based software systems, embedded systems, computer architecture, design automation, hardware design, and digital application-specific integrated circuit design. He received his BS in electrical engineering from the Univ. of Pittsburgh, his MS in electrical engineering from Purdue, and his MS and PhD in electrical and computer engineering from CMU. Contact him at Inmedius, Inc., 417 S. Craig St., Pittsburgh, PA 15213; jbeck@inmedius.com.

Jim Jennings works on behalf of IBM in several international standards organizations. His research interests are in embedded computing. He received his PhD from the Computer Science Department of Cornell University. Contact him at IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532; jsj@acm.org.