Opportunistic Annexing for Handheld Devices: Opportunities and Challenges

Jeffrey S. Pierce, Heather Mahaney, Gregory D. Abowd
Georgia Institute of Technology
{jpierce, mahaneyh, abowd}@cc.gatech.edu

Handheld devices, whether personal digital assistants (PDAs), cellphones, or something else, currently have limited processing, storage, input, and output capabilities. The first two, processing and storage, should continue to improve as advances in semiconductor technologies bring us faster and smaller chips. The built-in input and output (I/O) capabilities are less likely to improve because handhelds must remain small enough to easily slip into a pocket. Improving the I/O capabilities of handhelds is an important research challenge.

Most of the current research in this area focuses on making better use of handhelds' existing I/O capabilities. A much less common approach is to explore how handhelds can make use of additional devices. We believe that the increasing density of I/O devices we encounter in our daily lives presents an opportunity to improve the I/O capabilities of handhelds through *opportunistic annexing*.

Opportunistic annexing is the process of temporarily attaching devices to a computational environment (consisting of one or more devices such as a laptop, PDA, cellphone, etc.) to enhance its capabilities. Increasing an environment's processing power and storage capacity through grid computing and networked storage, both hot topics in the research community, are examples of opportunistic annexing.

Although annexing I/O devices has received less attention, researchers have started exploring how devices can combine their I/O capabilities to offer a more effective user interface. Example projects include Pebbles [3], an architecture that allows handhelds and desktops to work together, and Augmented Surfaces [9], a hybrid computing environment that allows devices to function as part of a larger whole.

In general these projects have focused on annexing handhelds to improve the interaction with desktop or wall-based environments. We are interested in the opposite: how users can annex parts of desktop and wall-based environments to improve interaction with their handheld. This direction has received less attention from the research community. A notable exception is the Personal Server [12], a handheld computer that relies almost entirely upon opportunistic annexing for I/O.

Our goals in this paper are to convince researchers that opportunistic annexing can address the limited I/O capabilities of handhelds and to lay out some of the challenges for opportunistic annexing. We will discuss approaches to generating applications of

opportunistic annexing and describe some of the challenges to designing the user experience. We will also discuss potential architectures for implementing interfaces and describe some of the security challenges.

GENERATING POTENTIAL APPLICATIONS

Most of the obvious applications of opportunistic annexing enhance traditional uses of handhelds. Users might annex a keyboard to type a memo into their PDA or a monitor to see their schedule for the month without scrolling. While we believe these applications will be useful, they are only a small part of the design space. Other approaches to thinking about the design space can suggest further applications.

Consider Benefits

One approach is to consider the potential benefits. Speed, a traditional metric for the effectiveness of an interface, is one potential benefit. Annexed input devices might increase the input rate, while annexed output devices might allow users to save time by eliminating the need to scroll or to switch views. Another potential benefit is a higher quality user experience. Annexing a display with a larger color palette or a higher resolution could yield this benefit. Reduced power consumption is another potential benefit. On planes a users could annex the seat display in front of him to read an electronic document, allowing his handheld to stop driving its display. A broader potential benefit is the addition of new functionality. Users could annex a printer to create hardcopy, annex a phone handset to capture a conversation, or annex a display while talking on their cellphone to allow them to simultaneously talk and view their schedule.

Consider Devices

Another approach is to consider devices that users might annex. The most interesting ideas may come from considering I/O devices beyond the standard keyboard, mouse, and monitor. Sample devices include televisions, projectors, microphones, speakers, digital pens, cameras (still or video), game controllers, phones, and hardware sensors. One of the most cited examples of annexing is annexing a projector to give a talk from a handheld [3,4]. Another possibility is for users to annex speakers to play music from their handheld. Users could also annex a borrowed digital pen to generate both hardcopy and digital versions of a jotted reminder.

Consider Relationships

Potential relationships between users and annexed devices can also suggest applications. We expect that a common situation will involve a user annexing her personal devices to accomplish a simple, light-weight task. A user might, for example, annex her keyboard, mouse, and part of her desktop monitor to quickly cut and paste directions to some event from her desktop to her PDA. Users might annex devices owned by their organization to share information or give a demo to a colleague. When visiting a colleague's office, for example, a user could annex a monitor to show research results. User might also annex a display as a guest or visitor. For example, a user at an airport might annex a public terminal to quickly search for the phone number of the person meeting her.

Consider Ratios

Considering the ratio of producers to consumers can also suggest applications. In addition to a user working alone (a 1:1 ratio), multiple users might annex a single input device to allow a single producer to serve multiple consumers. For example, users could annex the pen of a meeting scribe to capture a copy of his notes. The reverse, multiple producers and one consumer, is also a possibility. By annexing a display visible only to the lecturer, for example, a class could submit questions from their handhelds. A final case is multiple producers and consumers. Consider a pair of users who want to locate a time when they can both meet. They might annex a monitor and overlap their schedules on it to find a suitable time.

DESIGNING THE USER EXPERIENCE

After identifying potential applications we must design the user experience: the process of annexing devices, the interfaces to applications, the process of releasing devices, and the process of granting annexing permissions.

Annexing Devices

Users may annex devices either directly or through an intermediary. We anticipate that in the short term handhelds will annex most devices indirectly using the computer they are attached to as an intermediary. The drawback to this approach is that users must trust the intermediary not to eavesdrop on or interfere with their communications with annexed devices.

In the future we expect that users will be able to directly annex devices. For example, a user might borrow a keyboard and connect it directly to his handheld rather than through the computer it is normally connected to. This capability will place new processing, storage, and connectivity demands on annexable devices. At a minimum these devices will need to be able to set up a communications channel with the annexing device, and they should also be able to encrypt and decrypt communications with that device.

A primary issue for annexing devices is whether the user annexes devices using his handheld or using the devices he wants to annex. The former can be thought of as "pushing" the interface out to other devices from the handheld and the latter as "pulling" the handheld interface out using other devices.

Annexing via pushing. Annexing via pushing is better for both privacy and security. Pushing provides more privacy because the handheld does not need to keep other devices aware of its presence, reducing the risk of people tracking a user by monitoring the presence of his handheld. Pushing eliminates the risk of unauthorized connections and removes the processing and power burden of distinguishing valid and spurious connection requests. The primary disadvantage of pushing is that the user needs direct access to the handheld; this approach is not possible if the handheld is in the user's backpack.

When pushing, the quickest and simplest method of selecting a device to annex may be to physically specify the desired device by scanning an attached tag, such as an RFID tag or

bar code, with the handheld. Devices with a tag would be easily identifiable as devices available for annexing. A slower approach would be to allow users to poll for available devices. The resulting list of devices must use names that users can easily match to devices in the environment; one approach would be to print the names on the devices. A still slower approach would be to make users manually enter the name of the device they want to annex.

Once the user has a name for the device to annex, whether that name is a text description or an ID from a tag, the next step is to match the name to a network address. If the user queried for available devices, the query result could contain both a name and an address for that device. Otherwise the handheld will need to find the network address associated with a particular name. Unless the device's name is its address, the handheld will need to poll for the device by name either directly or through a name server.

Annexing via pulling. Annexing via pulling allows users to annex devices without interacting with their handhelds. For example, a user talking on a cellphone might want to use a nearby display to view his calendar without interrupting his phone conversation; pulling would make this possible. The primary drawback of annexing via pulling is that it forces users to authenticate themselves to their handhelds when annexing a device.

An advantage of pulling is that users already know the name of their handheld; they just have to determine its address using a method we described. However, while annexing via pushing should be equally straightforward for both directly and indirectly annexed devices, annexing via pulling is likely to be more difficult for directly annexed devices. Consider a user trying to attach a digital pen to her handheld without the benefit of an associated display. The user could perhaps write a special control character followed by the name of the handheld, but providing feedback that the pen attached successfully would be problematic.

Regardless of whether users annex devices via pushing or pulling, the process must be sufficiently quick and easy that annexing adds value. If a user could save 5 seconds by typing rather than writing a note on their PDA, annexing the keyboard must take less than 5 seconds to be worth the user's time. One approach to streamlining the process would be to cache the names and addresses of frequently annexed devices, particularly if we could associate that information with particular locations.

Designing User Interfaces

Designing user interfaces when users can opportunistically annex devices presents two particular challenges: learning how to effectively divide interfaces across annexed devices and learning how to design interfaces given uncertainty about the devices users will employ.

Dividing Interfaces. With the exception of research on computer-supported cooperative work, where researchers have done some exploration into effectively spreading interfaces across multiple devices for multiple users, most existing interfaces presume a single display and one or two input devices. Because opportunistic annexing allows users to

employ multiple I/O devices, we must explore how to spread the interface across these devices to utilize them most effectively.

Researchers have created several point designs. One of the best examples is Pebbles' Slideshow Commander [3], which allows users to control a slideshow and view notes on a handheld while displaying the actual slides using a projector. This configuration allows a presenter to control her talk without standing near a keyboard and to refer to her notes without carrying hardcopy.

One of the questions we will need to answer is whether we should treat multiple displays as separate spaces or as a single workspace. For some applications the decision may be clear cut, but for others either choice may be viable. Consider a program that allows users to show pictures on an annexed monitor. One way to structure the interface would be to separate the display spaces so that the user selects an image on the handheld and it appears on the monitor. Another way would be to treat the two as a single workspace and make the user drag images between the displays. To determine which is better we may need to build both and evaluate them.

We will also need to learn how users will want to divide an interface to protect their privacy. Given that people are more likely to read information on large displays than small ones [10], users may be reluctant to show certain information on large, public displays. We recognize that an interface designed to protect a user's privacy may not be the most effective in terms of speed or space, but an application that shows a user's social security number on a large public display will probably not be acceptable to users.

To address privacy concerns we could make users explicitly specify what information their handheld can and cannot display on annexed devices, but we would prefer to avoid burdening the user in this way. We suspect that we can learn some general heuristics about what information users are willing to display on other devices. For example, a user may not mind annexing a large display to view his bank statement while alone in his office, but would never want the same information shown on an identical display in an airport. We would need to allow users to override those heuristics if they do not fit the current situation. We could also try an adaptive approach, attempting to learn what information a particular user considers private and how that user feels about displaying that information on different devices in different situations.

Designing Under Uncertainty. The other challenge is learning how to design interfaces given uncertainty about which I/O devices users will employ. We need a method of creating interfaces that address this uncertainty. Two currently used methods are handcrafting interfaces for a variety of devices in advance and generating interfaces at run-time.

Designers may choose to handcraft the desired interface, creating different implementations for a variety of possible devices, because handcrafted interfaces tend to be easiest to use. This approach can work well when designers can accurately forecast

what device combinations users are likely to employ, but it has problems when the set of likely combinations is very large.

Some researchers have explored automatically generating interfaces (e.g. XWeb [6]). They provide a semantic description that avoids specifying layout or types of input. Devices can use this description to create an interface at run-time that fits their abilities. While this approach allows interfaces to adapt to a variety of I/O devices, it does not guarantee that the interfaces will be aesthetically pleasing or easy to use.

We expect that in practice users will primarily annex desktop displays, keyboards, and mice. Rather than choosing one approach or another, a more effective method might be to craft interfaces for common cases and rely on generated interfaces when users annex unanticipated devices. The iCrafter system uses this method [5]. Another possible method would be to allow designers to craft sub-components of interfaces but leave their assembly to the handheld at run-time.

Closing the Connection

A final design issue is determining how long devices remain annexed. In some cases it may make sense to allow users to only annex devices for a single, discrete action. For example, a user annexing a printer in order to print a paper will not need the printer again. In other situations we might allow users to annex a device for an unlimited amount of time. Between these extremes, we might allow users to "lease" annexed devices for a specified amount of time. The appropriate solution will depend on a variety of factors: the desired use, the user's identity, who owns the annexed device, etc.

If we allow users to annex a device for an unlimited amount of time, we will need to provide a mechanism for users to explicitly or implicitly release annexed devices. While users should be able to explicitly release indirectly annexed devices using either the handheld or the intermediary, they will need to explicitly release directly annexed output devices using the handheld. Releasing directly annexed input devices will probably be easiest through the handheld; otherwise users might need to remember a special input sequence to perform with the device to release it. Supporting the latter case may be necessary to allow users to release devices without retrieving their handheld. Users could implicitly release a device by moving away from it. Handhelds could use attenuation of the network signal or location information, if available, to attempt to determine when the user has moved away from the device.

Handhelds will need to retrieve any modified data when they release a device. While they can incorporate this step into the release procedure, they will also need a mechanism to prevent lost data when they unintentionally release devices, for example when an intermediary crashes. One possibility is to periodically update the handheld with any changes. The update frequency will need to balance increased power consumption against the risks of losing changes.

Granting Access

Users annexing devices are only part of our user population; we must also determine how device owners should specify who can annex their devices. Owners will need to be able to specify which of their devices users can access and what permissions different users have on those devices. Owners should be able to assign permissions to both individuals and groups of users.

Controlling access to indirectly annexed devices should be the easiest. Owners can use the intermediary computer to set the permissions for attached devices. The process will be more difficult for directly annexed devices because owners will first need to prove that they own the device; otherwise any users annexing those devices could modify the permissions.

As their collection of devices grows, owners may want a more centralized method of specifying who can annex which devices. For example, the computer support staff for a university might want to be able to set the permissions for all of the devices the university owns simultaneously from a single location, rather than assigning permissions to each device separately.

We must also determine the appropriate levels of granularity for owners to use when setting permissions. For example, the owner of a printer might want to be able to specify different page quotas for different classes of users, while the owner of a display might want to be able to specify different lease lengths and maximum window sizes for different individuals. Real world experience will be needed to determine the types and values of parameters that owners should be able to specify.

Evaluation

We need to demonstrate that handheld users will derive value from opportunistically annexing I/O devices before we expend the time and effort necessary to create a complete infrastructure. While we can make a priori arguments for the value of some applications (see sidebar), we believe the best way to demonstrate value is to create a variety of applications that simulate opportunistic annexing and formally evaluate them. For tasks that users can perform with a handheld alone, these evaluations will attempt to establish that opportunistic annexing allows users to complete those tasks more quickly. For new tasks that annexing makes possible, these evaluations will attempt to establish that annexing allows users to effectively complete those tasks.

Consider a user who wants to annex a display to consult his schedule while using his cellphone. In theory this application seems worthwhile: users can set up a meeting with the person they are talking to without interrupting the conversation to manipulate their schedule. By creating a simulated version of this application we can study users completing this task with and without the annexed display and measure whether the annexed display reduces both task completion time and the number of disruptions in the conversation.

In addition to demonstrating the potential value of annexing, these evaluations will help us establish constraints for the infrastructure. If users save ten seconds by annexing a monitor to set up a meeting on their cellphone, annexing the display must take less than ten seconds in order to be useful for that task. We will evaluate candidate infrastructures to verify that they fulfill the constraints.

We will eventually need to study the use of opportunistic annexing with handhelds over a longer time period. A longitudinal study could verify that users annex devices after the novelty wears off and identify which applications users find particularly useful. Unfortunately, without a fully deployed infrastructure a longitudinal study is likely to show that users do not derive value from opportunistic annexing because they do not have enough to annex. A useful intermediate step may be to build a small test bed and study use within that environment.

ARCHITECTURES FOR INTERFACES

We will need a software architecture that allows us to implement the user experience we design. We can classifying the alternatives by taking the Model, View, Controller model of user interfaces, along with the associated data, and considering how we might apportion the components among the available devices.

Deciding where the Controller lives should be straightforward. We expect that when the user directly annexes devices the Controller will live solely on the handheld. When the user annexes devices through an intermediary the Controller could shift to the intermediary, remain on the handheld, or divide its functionality between the devices. In any case, devices should transmit their event data directly to the Controller. Less common devices, such as cameras, could preface the event stream with a semantic description of the data. This description could, for example, communicate the data type, image size, and update rate for an annexed video camera.

Deciding where the other components should live is more complex because there are several valid possibilities. One approach is to send the View to the display by sending the actual pixels; an annexed display can copy them directly into its display buffer. This approach puts the burden primarily on the handheld. The handheld must be aware of the display capabilities of annexed devices, and it must be able to render and transmit the desired View at interactive rates. The advantage of this approach is that it is simple enough to work with almost any device. The drawback is that the processing, storage, bandwidth, and power demands of complex interfaces will exceed the capabilities of today's handhelds.

A second approach is to send semantically described data and rely on the display to generate the View. XWeb uses this approach [6]. The key to this approach is to determine the appropriate semantic level to describe the data. Higher-level descriptions allow displays to create higher-quality Views, but displays are less likely to recognize an arbitrary high-level description. Handhelds could try to balance this trade-off by providing both an optimistic, high-level description of the data and a more conservative description. Consider a user who wants to show her calendar on an annexed display. If

the display knows how to visualize a meeting, the handheld can send the data described as a collection of meetings. Otherwise the handheld could send the data as a collection of more common data types (e.g. numbers, dates, text strings).

Another approach is to send the data and a semantic description of the desired View. Cooltown [2] and the Personal Server [12] use this approach, employing HTML to describe the interface. This approach requires balancing the level of the description with the interface elements the display supports; telling a display to show meeting data using a calendar widget will not help if the display has no idea what a calendar widget is. Providing both optimistic and conservative descriptions may also be optimal for this method.

A final approach is to send a Model and View in the form of code along with the data. This approach provides as much flexibility as sending the pixels to display, plus it shifts most of the burdens from the handheld to the annexed device. The drawbacks are that every device must be able to execute the code and that it introduces the risk of malicious code. Both Jini [11] and SpeakEasy [4] use code to transfer interfaces between devices.

Because the utility of opportunistic annexing will depend on whether encountered devices use the approach as handhelds, handhelds may need to support multiple approaches. For example, a handheld might employ mobile code where possible to create a custom interface on an annexed display, but fall back on a semantic description of the interface where necessary. Further exploration is needed to determine the best choices for approaches.

SECURITY

We will need to provide mechanisms to protect users' devices and data. While current security protocols should be adequate for securing connections between devices, we will need to explore mechanisms to authenticate users, establish trust between users and annexed devices, and guard against unauthorized accesses.

Authentication and Trust Management

Users who want to annex a device must prove that they have the right to do so. In addition, users must establish a level of trust with the device. This trust goes both ways. Annexable devices need to be able trust users not to use them inappropriately, and users need to be able to trust annexed devices not to capture their data or access their handheld inappropriately. Together these requirements highlight the need for an authentication and trust management process [1].

While passwords, a form of credential, are the most common authentication mechanism, they are one-sided: they verify that the user has the right to access the device, but they say nothing about whether the user can trust the device with his information. A better mechanism might be to use signed certificates as credentials. By issuing certificates to both users (or more accurately their handhelds) and devices, we can allow devices to authenticate users and users to authenticate devices. For example, a university student might have a certificate from the university verifying that she should have student-level

access to devices, while a monitor in one of the university's public labs might have a certificate verifying that the monitor is owned by the College. If the user asks to annex the monitor to display some information, the monitor can verify that the user, as a university student, has the right to do so, while the user can verify that the monitor is actually owned by the university.

Streamlining the authentication process, while keeping it reliable, will be a primary concern. While devices can verify credentials themselves, a more common approach will probably be to rely on a separate authentication server, which can introduce delay. Creating dedicated keys for encrypting and decrypting communications with each other, once devices have verified credentials, might help streamline this process. The user's handheld can encrypt future annexing requests to that device with its key; successfully decryption using the corresponding key will authenticate the user.

When annexing via pulling, users will have to perform an additional step: proving to their handheld that they are the ones initiating the annexing request. Unfortunately, certificates will not work and passwords are subject to replay attacks: a borrowed keyboard could store the keystrokes, capturing the user's password and allowing an adversary to reuse it. Researchers have been working on new authentication methods (e.g. [9]), but more work is necessary.

The established level of trust will dictate how the user can employ the annexed device. However, users do not have to adhere to the maximum level of trust; they can make their own determination of how much to trust a particular device. For example, a user could have the same permissions when annexing his personal monitors, his organization's monitors, and the monitors in a local coffee shop, but he might trust those devices differently. He might be willing to send his actual data to monitors he owns, while for monitors owned by his organization he might prefer to render a view of his data on his handheld and send the view. For the monitors owned by the coffee shop he might prefer to avoid sending sensitive information at all. When viewing his schedule, for example, the user could display the blocks of time where he is busy on a coffee shop monitor while displaying the event details on his handheld.

We will need to provide some mechanism for users to specify a different level of trust than the one they are granted. We might be able to algorithmically suggest an initial level of trust given information about the sensitivity of the information and the relationship between the user and the device owner. However, this algorithm would probably need to err on the side of caution and provide a simple mechanism for users to override it.

Protecting Devices

Protecting the data on handheld devices from unauthorized access is a primary concern. Consider a user who annexes the I/O peripherals of a nearby desktop computer, using it as an intermediary, to look at his schedule for the day. The intermediary computer might be able to impersonate the user by pretending that the user has requested his weekly and monthly schedule as well, giving it access to that information.

One possible solution is to make users approve all data requests. While this approach allows users to prevent unauthorized access, approving every request may quickly become too cumbersome. A similar approach would be to allow users to specify the data that annexed devices can access during a particular session, but this approach might also be too cumbersome. Another approach would be to structure user interfaces so that users perform any actions that initiate a data transfer from the handheld, but this approach might prevent the use of the most effective interface designs.

A better solution might be to allow users to monitor communications with annexed devices using their handhelds. Monitoring could allow users to detect when unauthorized accesses occur and take the appropriate action, provided we can effectively communicate how the actions (data transferred, applications invoked, etc.) taken by an annexed device match the expected and allowed actions.

We will need to protect annexed devices as well. Owners of devices will doubtlessly be reluctant to allow annexing if there is risk involved. Software architectures that transfer and run code on annexed devices will probably be the most risky; owners might choose to prevent all but the most trusted users from employing mobile code.

CONCLUSIONS

We believe that opportunistic annexing can address the limited I/O capabilities of handheld devices. Although the idea is promising, much work remains. We are currently engaged in the first step: creating applications that simulate opportunistic annexing and evaluating them to demonstrate value. We hope that other members of the research community will join us in this effort and in addressing the other challenges we laid forth in this paper.

References

1. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, D.A. (1999) The Role of Trust Management in Distributed Systems Security. In *Secure Internet Programming: Security Issues for Mobile and Distributed Objects,* Jan Vitek and Christian Jensen editors. Springer-Verlag, New York.

2. Kindberg, T. and Barton, J. (2001) A Web-Based Nomadic Computing System. *Computer Networks*, 35(4), 443-456.

3. Myers, B.A. (2001) Using Handhelds and PCs Together. *Communications of the ACM*, 44 (11), 34-41.

4. Newman, M.W., Izadi, S., Edwards, K., Sedivy, J.Z., Smith, T.F. (2002) User Interfaces When and Where They are Needed: An infrastructure for Recombinant Computing. P*roceedings of UIST '02*, 171-180.

5. Ponnekanti, S.R., Lee, B., Fox, A., Hanarahan, P., Winograd, T. (2001) ICrafter: A service framework for ubiquitous computing environments. *Proceedings of Ubicomp '01*.

6. Olsen, D. R., Jefferies, S., Nielsen, T., Moyes, W., Fredrickson, P. (2000) Cross-modal Interaction using XWeb. *Proceedings of UIST '00*, 191-200.

7. Olsen, D. R, Nielsen, T., and Parslow, D. (2001) Join and Capture: a Model for Nomadic Interaction. *Proceedings of UIST '01,* 131-140.

8. Pering, T., Sundar, M., Light, J., and Want, R. (2003) Photographic Authentication through Untrusted Terminals. *IEEE Pervasive Computing*, 2(1), 30-36.

9. Rekimoto, J. and Saitoh, Masanori. (1999) Augmented Surfaces: A spatially continuous workspace for hybrid computing environments. *Proceedings of CHI '99*.

10. Tan, D. S. and Czerwinski, M. (2003) Information Voyeurism: Social Impact of Physically Large Displays on Information Privacy. *CHI 2003 Extended Abstracts*.

11. Waldo, J. (1999) The Jini Architecture for Network-Centric Computing. *Communications of the ACM*, 42 (7).

12. Want, R., Pering, T., Danneels, G., Kuman, M., Sundar, M., Light, J. (2002) The Personal Server: Changing the way we thing about ubiquitous computing. *Proceedings of UbiComp '02.*

Biography

Jeffrey S. Pierce is an Assistant Professor in the College of Computing and the GVU Center at the Georgia Institute of Technology. His research interests include personal information environments, 3D interaction, and interfaces to mobile devices. Jeff received his Ph.D. in computer science from Carnegie Mellon University. He is a member of the IEEE Computer Society and ACM. Contact him at jpierce@cc.gatech.edu.

Heather Mahaney is a Ph.D. student at the Georgia Institute of Technology. Her research interests include personal information environments and ubiquitous computing. Heather received a B.S. in computer science and mathematics from Washington and Lee University. Contact her at mahaneyh@cc.gatech.edu.

Gregory D. Abowd is an Associate Professor in the College of Computing and the GVU Center at the Georgia Institute of Technology. He is also the director of the Aware Home Research Initiative at Georgia Tech. His research interests involve applications research in ubiquitous computing, concerning both human-computer interaction and software engineering issues. He received a D.Phil. in Computation from the University of Oxford. He is a member of the IEEE Computer Society and ACM. Contact him at abowd@cc.gatech.edu.

Overcoming Limitations of Handheld Devices (Sidebar)

Annexing a larger display should improve interaction with handhelds. According to one study, reading speed increases 25% when document width is increased to full screen (18.7 cm) from a third of the screen (6.2 cm, about the width of a PDA) [2]. Another study suggests that smaller displays force users to reread text more often [1]. Users reading a document on a 20 line display jumped in the text three times more often than users with a 60 line display. Small screens also appear to make it more difficult to search for information [3]. Users completing four question-based search tasks on a 30 line display answered twice as many questions correctly as users with a 15 line display.

Annexing a keyboard should improve input rates for handhelds. The three predominant input techniques for PDAs are soft QWERTY keyboards, transcription, and Graffiti, while multi-tap input predominates for cellphones. In their summary of text entry for mobile devices [4], MacKenzie and Soukoreff reported input rates of 20.2 words per minute (wpm) for novices using soft QWERTY keyboards and estimated that experts could reach 30 wpm. They found transcription rates average 17 wpm. Palm claims experts can attain input rates around 30-35 wpm for Graffiti. Estimates for multi-tap input rates for experts on cellphones range between 20 and 27 wpm [5]. In comparison, average touch typists input between 40 and 60 wpm on a regular QWERTY keyboard, and experts range between 80-100 wpm. While researchers have developed a number of new input techniques for PDAs and cellphones, these techniques have yet to rival keyboards.

1. Dillon, A., Richardson, J., McKnight, C. (1990) The Effects of Display Size and Text Splitting on Reading Lengthy Text from Screen. *Behaviour and Information Technology,* 9*,* 215-227.

2. Duchnicky, R.L., Kolers, P.A. (1983) Readability of Text Scrolled on Visual Display Terminals as a Function of Window Size. *Human Factors*, 25, 683-692.

3. Jones, M., Marsden, G., Nasir-Mohd, N., Boone, K. (1999) Improving Web Interaction on Small Displays. *Computer Networks*, 31, 1129-1137.

4. MacKenzie, I.S., Soukoreff, R.W. (2002) Text Entry for Mobile Computing: Models and Methods, Theory and Practice. *Human Computer Interaction, 17,* 147-198.

5. Silfverberg, M., MacKenzie, I. S., & Korhonen, P. (2000). Predicting text entry speeds on mobile phones. *Proceedings of CHI 2000*, 9-16.