Please note: an updated version of this TR is available as MSR-TR-2002-87

Sideshow: Providing Peripheral Awareness of Important Information

JJ Cadiz, Gina Danielle Venolia, Gavin Jancke, Anoop Gupta

September 14th, 2001

Technical Report
MSR-TR-2001-83

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Sideshow: Providing Peripheral Awareness of Important Information

JJ Cadiz, Gina Danielle Venolia, Gavin Jancke, Anoop Gupta
Microsoft Research, Collaboration & Multimedia Group
One Microsoft Way, Redmond, WA 98052 USA
{jjcadiz; ginav; gavinj; anoop}@microsoft.com

ABSTRACT
A fundamental issue with user interfaces is how to help users stay aware of information without being overly intrusive or distracting. In this paper we describe Sideshow, a peripheral awareness interface designed to help users stay aware of people and information. We present data from a field trial of Sideshow where several hundred employees within our company used Sideshow over a seven-month period of time. The data indicate that Sideshow's design accomplishes the goal of providing awareness of important information without being overly distracting.

Keywords
Situational awareness, peripheral awareness, awareness, computer mediated communication, information overload

1 INTRODUCTION

As the world becomes increasingly asynchronous, digital, and distributed, keeping track of all the pertinent information in our lives has become incredibly difficult. As we work, documents are updated, information on web sites is modified, databases are changed, and the people we depend on come and go. It's largely believed that maintaining awareness (defined as an "understanding of the activities of others, which provides a context for your own activity" [3]) of all this information can be extremely helpful for productivity, especially for teams that have to work at different times and in different locations [4].

Sideshow is designed to help people maintain awareness of important information in their environment. Designing these types of interfaces involves a myriad of difficult tradeoffs centering on the fundamental limits of human attention. Many of these tradeoffs are discussed in previous literature, which we cover in the next section. We then discuss our design principles and Sideshow's design in section 3. To test our design, we deployed Sideshow within our company for several months in an ongoing field study. Section 4 details the results of this study, including data that evaluate our design and provide insights about possible future directions for awareness interfaces.

2 RELATED WORK

Researchers have pursued several strategies to address the problem of how to keep people aware of important information. We believe these strategies generally fall into one of three categories: polling, alerts, and peripheral awareness.

2.1 Polling Interfaces

The first way to help people stay aware of information is to make it accessible somewhere and allow people to repeatedly check—or "poll"—the information. If I want to find out the status of a file, I can open it. If I want to see my company's current stock price, I can visit a web site. If I want to see if one of my co-workers is available to talk, I can walk over to their office and check.

While this strategy is reliable and straightforward, it has a few obvious drawbacks. First, polling only provides information updates when users poll, and because important events often occur when users are doing other things, it's easy to miss critical updates. Second, polling often imposes a high memory burden on users: not only do they have to remember to poll, but they often have to mentally compute what has changed. Third, polling takes a tremendous amount of time and energy, especially if the information is spread widely throughout several files, web sites, etc. Several portal interfaces (like the My Yahoo! Portal page and Microsoft's Digital Dashboard) have successfully addressed this third drawback by creating a single interface that fetches information from multiple sources and summarizes it one place, but these interfaces still leave users to cope with the other failings of polling interfaces.

2.2 Alerts

A serious drawback of polling interfaces is that if a critical event occurs, users find out only when they poll the information source, which can be quite a while after the event has occurred. One solution is to interrupt users when something important happens. We call any interface that intentionally interrupts users an alerting interface. Examples of alerts include fire alarms, reminder windows that pop up in the middle of your screen (to tell you, for example, that you have a meeting in 15 minutes), and the sound that's played when you receive a new mail message.

Unfortunately, the primary strength of an alert is also its primary weakness: alerts require that users get interrupted from their primary task, and studies have highlighted the harms of these interruptions [2, 12]. In the extreme case, users can get interrupted by alerts so much that they can't get any work done. For this reason, several researchers have focused on two critical questions for alerting

interfaces: how should we interrupt users, and when should we interrupt users? Alerts can be delivered via audio or visual cues, can be delivered in both highly or minimally intrusive ways, and can be delivered using intelligent algorithms to determine if the cost of interrupting the user with an alert is worth the benefit. For example, Horvitz [10] has examined the use of Bayesian statistics to make cost/benefit judgments of interrupting users with alerts, based on the content of the alert and what the user is currently doing.

However, even if alerts are given at the best possible time and using the best possible method, by their very nature, they still must interrupt people. This presents a problem when trying to use alerts for many types of information, simply because you may want to stay aware of some information but never be interrupted by it, and because of the scalability problem: even if you want to stay aware of 100 pieces of information in one day, you probably don't want to receive 100 interruptions, regardless of their timing and method.

2.3 Peripheral Awareness

The polling and alerting strategies utilize our focal attention, but the third strategy takes advantage of our innate ability to stay aware of things in our periphery. This strategy, peripheral awareness, works by filling users' peripheral attention with information such that it envelops them without distracting them. With this method, the goal is to present the information such that it works its way into users' minds without intentional interruptions.

An excellent example is knowing what the weather is like outside. If you work in a window office, you likely have a very good idea about the current weather conditions. But how did that information get into your head? Does the weather interrupt you every time there's a change? Do you consciously look out the window and check the state of the weather every few minutes? Because the information persistently resides in your peripheral attention, it works its way into your knowledge and understanding of the world.

The promise of peripheral awareness has led to several interfaces and studies. Some research has focused on our ability to stay aware of things using peripheral audio [18, 19], but the majority of systems have focused on the use of peripheral vision or a mix of peripheral visual and audio cues.

One class of displays, ambient awareness displays, use the tactic of embedding information into users' surrounding environment, often without using standard computer screens. Perhaps the most famous example of an ambient awareness display is Weiser's twirling string that kept people aware of network traffic [22]. More recent work includes the Information Percolator, which utilizes water tubes and bubbles to display information [9], and the ambientROOM project, which has examined various ways of embeding information into various artifacts in a typical office environment [11].

Awareness interfaces can also utilize a more traditional screen-like secondary display [17]. In fact, sometimes the second display can simply be another monitor hooked up to the same computer. When studying people who used computers with multiple monitors, Grudin found that often the second displays weren't treated as more workspace, but as an area where important information could be displayed peripherally [7].

Several projects have also examined interfaces for providing awareness of information on users' main screen. Although these types of interfaces are more widely accessible than systems that require secondary displays, there's a clear issue of consuming users' precious screen real estate. Researchers have approached this issue in a variety of ways. Several studies have examined the use of tickers and faders, which have the advantage of being able to rotate through lots of information without taking up much space. Of course, a clear drawback is that tickers and faders are visually dynamic interfaces, and thus there's been some debate over how distracting they are [5, 14, 15].

Another approach has been to use interfaces that are gently blended into the background of whatever is currently on the screen [6, 8]. Still other techniques have involved simply creating an application that doesn't guarantee that it's always visible, but that is always running and available on the desktop [1, 16].

2.4 Using the Strategies Together

McFarlane [12] studied four different methods of interrupting people with information and found that no single method was the best. We believe the same is true for polling, alert, and peripheral awareness interfaces: each has its place, but the ideal interface will need to utilize all three strategies at the right times. If someone is trying to call you on the phone, you don't want an icon in your peripheral vision to change gently. Every time a stock you're interested in goes up or down, you don't want a klaxon to sound. And, of course, people shouldn't have to check a web page repeatedly to see if their building is on fire.

E-Mail as a Special Case

Sending updates via e-mail has emerged as one of the most popular methods for keeping people aware of information, perhaps because of the ubiquity of e-mail, and perhaps because of the ease of building a system that sends e-mail. However, when considering our strategies, e-mail emerges as a special case because the category it falls into depends on how people use e-mail. If users always keep e-mail open on their computer and check every message as soon as it comes in, then every e-mail is an alert. However, if users only open their mail once in a while to see what's happening, e-mail becomes a place to poll.

Unfortunately, using e-mail as the main interface for information awareness can result in inbox overloaded. One

difficulty of sending e-mail notifications is knowing when to send a message, and the result can be an inbox cluttered with notifications, some of which are no longer valid. E-mail interfaces are typically designed for person-to-person communication, not information awareness, and thus using e-mail for this purpose quickly creates problems.

3 SIDESHOW

Sideshow is an awareness interface with the goal of helping people stay aware of large amounts of dynamic information without overloading or distracting them. It resides on a user's primary display and utilizes peripheral awareness. We've also built Sideshow to support easy polling of information and have explored some use of alerts.

When designing an interface such as Sideshow, there are several design tradeoffs to consider. To describe Sideshow, first we'll outline our design principles and then we'll describe the interface.

3.1 Design Principles

The first design principle we followed when designing Sideshow was *make it always present*. Because we wanted Sideshow to utilize peripheral awareness, it was clear that we had to design the interface such that it was always present in the user's periphery when they were working on their computer.

Second, because we were building an interface that would always be in users' peripheral vision, our second principle was to *minimize motion*. Most of us have probably had the annoying experience of trying to read a web page with an animated advertisement on it. Because of the way our perceptual systems work, unexpected motion in our periphery tends to be highly distracting, thus we designed Sideshow to be as visually calm as possible.

The third design principle we followed was *make it personal*. There have been several high-profile commercial attempts at information awareness displays, two of the most visible being PointCast (now called Infogate) and Microsoft's ActiveDesktop. However, these commercial attempts haven't become overwhelming successes. We believe these interfaces failed not necessarily because of their design, but because they focused on generic information: information that's useful to everyone, but only minimally. Typical examples are news, weather, and stock prices. Thus, we made sure that Sideshow focused on information that was personally relevant and important for people to stay aware of.

Because of this focus on personally relevant information, and because no single company or organization can possibly own—or even know about—all the different types of information that people need to stay aware of, our fourth principle was to *make Sideshow extensible.*

Our fifth design principle was *support quick drill-down and escape*. Because of limited space in users' periphery and the potentially large number of items users want to stay aware of, peripheral awareness displays can't provide much detail about information the user is watching. However, information isn't very useful unless it's detailed, thus we designed Sideshow so that it would be very easy for users to drill-down to get highly detailed information. We also designed the drill-down mechanism to be easy for people to "escape" and return to what they were working on, in hopes of minimizing the costs of context switching.

Our sixth and last design principle was *make it scalable*. It's our feeling that people want to stay aware of a large number of information sources, thus we needed to design our interface to handle dozens of items.

Figure 1: The Sideshow sidebar. This sidebar resides on one edge of the user's desktop and always remains visible. The sidebar is filled with items we call tickets. Each ticket displays a small summary of information. If users want to find out more, they can hover their mouse over a ticket, which brings up a tooltip with detailed information.

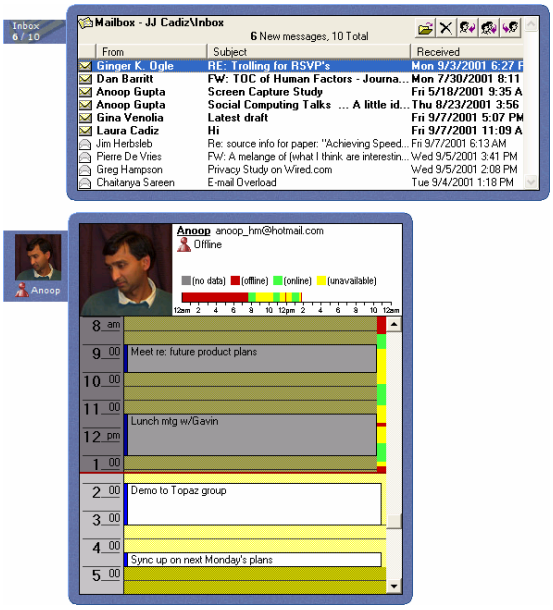

Figure 2: Two examples of tooltip grande windows. When users hover their mouse over the inbox ticket, the tooltip grande shows the contents of the inbox (top). Users can click messages to open them. When users hover their mouse over a person on the sidebar, they can see the person's calendar (if the person has made it available) and the history of when the person has been available and unavailable today using Windows Messenger (displayed as a color bar on the right of the calendar).

3.2 A Sidebar That's Always Present

Sideshow is implemented as a sidebar (see Figure 1). This sidebar, by default, is always present on one edge of the user's screen (much like the Windows taskbar). Users can also configure the sidebar to automatically hide itself or to allow itself to be covered by other windows. By default, the sidebar is 55 pixels wide. Inside the sidebar are several high-level summaries of important information in a user's world. We call these summaries "tickets."

3.3 Getting More Information

If users want to find out more information about a ticket, they can hover their mouse over it and a "tooltip grande" appears (Figure 2). We chose the name "tooltip grande" because the window behaves like a tooltip (it appears when you mouse over a ticket and disappears when you move the mouse away), but these tooltips differ from standard tooltips in two ways. First, they're rather large (to provide lots of detailed information), and second, they're actionable: users can manipulate information inside the tooltip window or click to get even more detailed information. For example, in the tooltip for my mail inbox, I can open, forward, reply to, or delete any message.

If users want even more information than the tooltip grande provides, they can double-click a ticket to bring up the source of the information. For example, double-clicking the Inbox ticket opens the user's inbox; double-clicking the MSFT stock ticket opens up a web page at moneycentral.msn.com with detailed information about Microsoft's stock price.

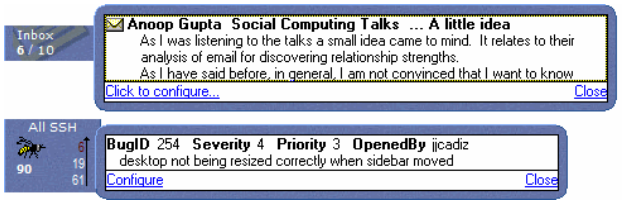

Figure 3: Two types of alert windows provided by Sideshow. When new mail arrives or when a bug of interest changes, a window fades in with a summary of the information. Users can click on the alert window to get more information.

3.4 Minimizing Motion & Exploring Alerts

Sideshow minimizes motion primarily by changing very few pixels when information is updated. For example, updating the information in several of the tickets involves just changing a few numbers, which affects relatively few pixels. For the more graphical tickets, updates only occur once every five or fifteen minutes, and if the images need to change, the changes typically aren't dramatic.

While we've taken great care to design tickets that are minimally distracting, as noted in section 2.2, people sometimes want to be interrupted by important information. Thus, we implemented alerts for two ticket types (see Figure 3). First, when new mail arrives in one of the inbox tickets, a window summarizing the new mail fades in and then fades out after a few seconds (users can also configure the window to persist until they explicitly dismiss it). The "My Bugs" ticket has a similar window that fades in and out any time a software bug of interest is added or modified. In both cases, users can click on these windows to read the new e-mail messages or see the changes to the bugs.

3.5 Making Sideshow Personal

As mentioned in section 3.1, a major focus of Sideshow was creating tickets that helped people stay aware of information that was critical to their work. While we developed some generic tickets as proofs of concept (stock, news, weather, etc.), our focus was on information that people often said they "lived" in. For example, in our company, Outlook is the main calendaring and e-mail tool, and people often say they "live" in Outlook. As another example, RAID is our company's internal tool for reporting and tracking bugs, and people often say they "live" in RAID. For these reasons, we spent considerable time developing tickets to watch Outlook Calendars, Outlook mail folders, and RAID bugs.

In addition, because many people work on highly interdependent teams, we also spent considerable time developing tickets that help people stay aware of their co-workers (similar to the interface described in [21]). As shown in Figure 1, individual co-workers can be placed on the sidebar and identified by static images; these images indicate whether a person is currently available or not. The tooltip for the person shows their calendar for the day (if

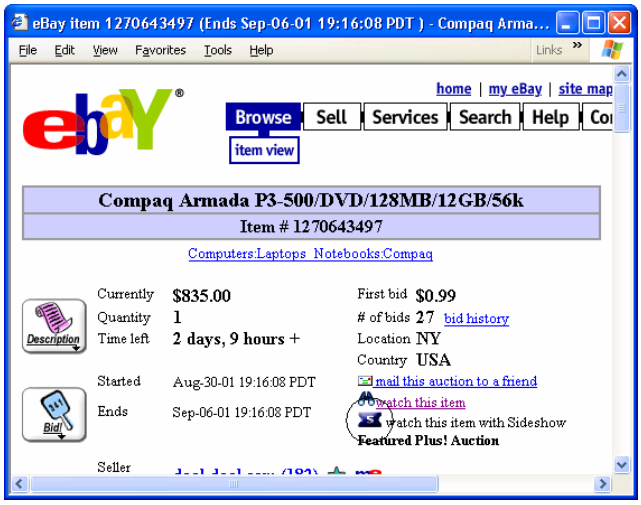

Figure 4: A mock-up of a ticket (circled) on a web page. Tickets can be placed on web pages, and users can drag these tickets to their sidebar to watch different types of information (an eBay auction, in this case).

they've enabled access to it), as well as when the person has been available and unavailable today according to Windows Messenger (Figure 2).

3.6 Making Sideshow Extensible

Although creating excellent Outlook and RAID tickets were a good step toward making Sideshow as useful as possible, we knew that we couldn't create all the tickets that would allow people to use Sideshow to watch all the important information in their world. Similar to the problems outlined by MacLean et al. in [13], several of the divisions within our company have custom tools and processes that made writing tickets for each of these divisions impossible. Thus, we followed a model similar to MacLean's: we provided the necessary tools and distribution processes such that one motivated person in a division could write incredibly valuable tickets and then distribute them to their division. First, we released a Sideshow SDK (software development kit) that allowed people to author tickets using HTML or C++. Second, we designed Sideshow tickets such that they could be distributed as files, which enabled people to send tickets by e-mail or post them on web pages (Figure 4).

3.7 Supporting Scalability

We implemented several features to support large numbers of tickets on the sidebar. First, tickets can be placed in groups and these groups can be collapsed or expanded. When a group is collapsed, its tooltip grande shows the tickets inside it, and users can mouse over the tickets in the tooltip grande to bring up another tooltip grande with more information about the ticket. Second, tickets resize themselves based on how much space is available on the sidebar: as long as there's enough space, tickets display themselves at their ideal size. However, once the sidebar fills up, tickets start shrinking until they reach their smallest possible size (ideal size and smallest possible size are determine for each ticket by the ticket developer). If another ticket is added once all tickets have reached their smallest possible size, the tickets at the bottom scroll off into an overflow area that's accessible by scrolling the sidebar. Scroll buttons (and a button that allows users to peek at the tickets in the overflow area via a tooltip) only appear on the sidebar when the mouse is over it (not shown in Figure 1).

4 FIELD STUDY

Although we designed Sideshow with several good principles in mind, many practical questions remained. Would people surrender ~50 pixels from the edge of their screens to run Sideshow? Would Sideshow be distracting? What were the tickets we needed to develop to make Sideshow as useful as possible? Would the mechanisms we designed for getting more information work well for users? We sought to answer these questions by deploying Sideshow within our company.

4.1 Methodology

Sideshow was first presented to our company during an internal demo festival held in January 2001. Afterward, approximately 200 people installed Sideshow. Although we made no effort to publicize Sideshow after this demo festival, word of Sideshow spread throughout the company, and by August 2001, we had nearly 2000 installations.

We collected data about Sideshow using two methods. First, we instrumented Sideshow such that many user interactions (adding tickets, bringing up tooltips for tickets, changing the width of the sidebar, etc.) were logged. Second, during August 2001, we released two surveys. The first survey was given to 860 people who were currently using Sideshow and asked about a variety of Sideshow's features. 309 people responded (a 36% response rate). The second survey was given to 698 people who had used Sideshow for more than three days but hadn't used Sideshow in the prior two weeks. This survey asked a small number of questions about why people stopped using Sideshow. 178 people responded (a 26% response rate). 275 people who downloaded Sideshow but did not use it for at least 3 days were not surveyed.

Before presenting results from this field study, it's important to note a few things. First, all of these data are from a self-selected user population. None of these users were randomly selected as participants. Second, all of these users work for our company and arguably have above average computer skills. However, these data are from several hundred users, some of whom used Sideshow for several months. In addition, our user population is quite diverse. Our users include administrative assistants, sales staff, finance staff, software developers, product designers, lawyers, product support professionals, and vice presidents. Furthermore, because our company has field offices throughout the world, these data are from people in over 20 countries throughout North America, South America, Europe, Asia, and Africa.

It's also important to note that we iterated extensively on Sideshow's design throughout this field study. While the basic concept of having a sidebar with tickets has remained the same, many bugs have been fixed and many features have been refined over the course of the study, which is typical for an iterative design project.

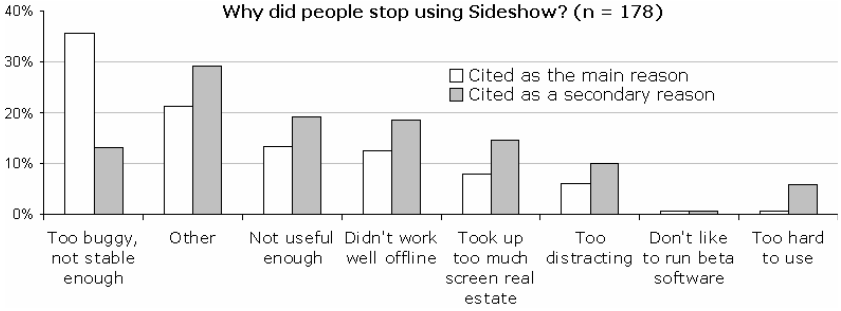

Figure 5: The reasons people said they stopped using Sideshow. People could select one primary reason and multiple secondary reasons.

4.2 Is Sideshow Useful?

Perhaps the most basic question for a new UI concept like Sideshow is whether it's useful. Although there are many ways to measure usefulness, because Sideshow takes up precious screen real estate and people had no incentive to use it, we look to adoption as a measurement of usefulness.

According to our usage logs, 1907 people installed Sideshow during our seven-month field study. We classified these users according to how much they used Sideshow: people who used Sideshow at least once in the past 14 days were considered **current users**, people who had used Sideshow for at least 3 days but hadn't used it in the past 14 days were considered **light users**, and everyone else (people who used Sideshow for less than 3 days) was considered a **one-time user**.

According to this classification, of the 1907 people who installed Sideshow, 238 (12.5%) are one-time users, 681 (35.7%) are light users, and 988 (51.8%) are current users (these numbers are slightly different from the numbers described in section 4.1 for the surveys because the usage analyses were performed at different times). Because people installed and used Sideshow at varying times, it's also important to look at the number of days that people ran Sideshow on their desktop. One-time users used Sideshow a median of 1 day (average of 13 days), light users used Sideshow for a median of 21 days (average of 37.2 days), and current users used Sideshow for a median of 33 days (average of 55.4 days).

Given these numbers, it seems safe to say that the majority of people thought Sideshow was a useful concept. When examining data from the people who had stopped using Sideshow, the most often cited reason for stopping was simply that the prototype was too buggy and not stable enough (see Figure 5). In fact, our biggest concern—that people wouldn't be willing to give up screen space to run Sideshow—was not supported. Only 8% of people cited this as the main reason they stopped using Sideshow (15% listed it as a secondary reason).

Furthermore, when we asked current users of Sideshow whether they thought it was worth giving up the screen space to run Sideshow, users' median response was 4 ("agree") out of a 5-point scale (see Table 1). In addition, although we allowed people to set their sidebars to disappear when the mouse wasn't over it ("autohide"), only 13% of current and light users chose to enable this feature. In fact, on average, people adjusted their sidebar to take up 7.3% of their screen space.

Overall, there was a very high level of excitement about Sideshow. Many people asked when it would become a product, and at the end of the summer, interns asked if there was any way they could take it with them back to college. Just a few examples of positive feedback include:

> *"I love Sideshow" (5 respondents)*
>
> *"I think it ROCKS!"(3 respondents)*
>
> *"It is an extremely helpful tool. Just running the mouse pointer over the ticket so it expands makes life so much easier than having to keep all different programs and windows open and browsing them."*
>
> *"Let me know when I can send it to my friends and family!!!"*

4.3 What makes Sideshow useful?

To determine what makes Sideshow useful, we examined a variety of usage and survey data. First, we asked all the people who were currently using Sideshow what the main reason was they continued to use it (Figure 6). 26% said they continued to use Sideshow because it made it easy to work with their calendar and e-mail. 20% cited other reasons, which mostly had to do with how Sideshow made it easy for them to stay aware of a variety of information. In particular, users seemed to like how Sideshow allowed them to stay aware of important information without switching away from their primary task. Users wrote:

> *"I like the quick glance to see amount of mail, bug status, traffic and other info without having to open 10 others apps to get the same info."*
>
> *"I love the way it allows me to track lots of information at the same time. I like so many tickets it's impossible to pin one down."*
>
> *"It's not just one ticket that makes it good, it's the fact that all the important info is presented in the smallest possible space."*

4.4 Is Sideshow Distracting?

One of the most significant worries of awareness researchers is creating a distracting interface. While we

Table 1: Selected questions from the survey of people currently using Sideshow (n = 309).

Question 1 = Strongly Disagree 5 = Strongly Agree	Median	Avg	Std Dev
Sideshow is distracting	2.0	2.3	0.9
Sideshow interrupts me when I'm trying to do other work.	2.0	2.2	0.9
It's worth giving up the screen space to run Sideshow.	4.0	3.8	0.9
Sideshow grabs my attention at the right times.	4.0	3.6	0.8
Sideshow helps me stay aware of information that's critical for me to keep track of.	4.0	3.7	0.8
I like being notified by Sideshow as soon as new mail arrives.	4.0	4.1	1.0
Sideshow's e-mail notifications often distract me from doing important work.	2.0	2.3	0.9

weren't able to perform any controlled lab studies to measure how distracting Sideshow is, we did ask questions about intrusiveness in our surveys.

First, we asked current users if they thought Sideshow was distracting. The median score was 2 ("disagree"). Second, we asked people who stopped using Sideshow whether they stopped because it was distracting. Only 6% said the main reason they stopped using Sideshow was because it was distracting, and only 10% cited it as a secondary reason.

Finally, we looked at what people thought about what we believe is the most distracting part of Sideshow: the alert windows that pop up when new mail arrives. The median response to the question, "I like being notified by Sideshow as soon as mail arrives" was 4.0 ("agree"). The companion question, "Sideshow's e-mail notifications often distract me from doing important work" received a median score of 2.0 ("disagree"). When asked about the e-mail alerts, users wrote:

> *"That's what I like best about Sideshow. Being able to see who the new mail is from and determine whether I should read it now or not..."*

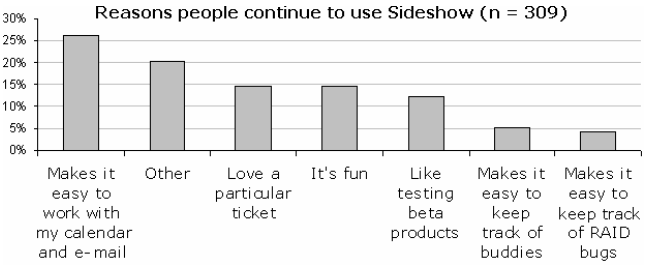

Figure 6: The main reasons people continue to use Sideshow.

> *"Love it - being able to do a quick scan to see whether it is an urgent email or not really helps in my role and saves the time previously taken checking Outlook when the new mail icon appears in the tool tray at the bottom of the screen."*

> *"The notification is useful because it prevents me having to go to Outlook as often. Most messages I can read later. Although the popup is distracting, I find its usefulness is worth it."*

> *"One of the best features in Sideshow"*

4.5 Areas for Improvement

In addition to all the positive feedback about Sideshow, users also provided lots of areas in which we could improve Sideshow. Several people made comments about wanting tickets that were more personalized for them. For example, the majority of the employees in our company are located in one region, and many of our tickets were focused on information in that region (traffic, weather, etc.). Many people outside this region expressed a strong desire for tickets that watched information that was relevant to their own region. Making Sideshow faster and more tolerant to times when the network wasn't available were also frequent suggestions, but these issues have more to do with Sideshow being a research prototype and less to do with Sideshow's core concepts.

5 CONCLUDING REMARKS & FUTURE DIRECTIONS

Sideshow continues to be used by hundreds of employees in our company. Potential future research directions include studies involving users outside our company, and studies of Sideshow on mobile devices. Tang's Awarenex project [20] examines interfaces and architectures for supporting awareness of important information on Palm and RIM Blackberry devices, and we believe this is a fruitful direction for Sideshow as well. We're specifically interested in putting Sideshow on mobile devices not just to provide people with awareness of their important information while away from the desktop, but also because mobile devices can serve as secondary peripheral displays when docked next to a user's primary screen.

We're also interested in methods to help users customize their sidebars without lots of effort. For example, if Sideshow notices that you visit a document that's often edited by other people, it could place a ticket to watch the document in a "recommended tickets" group on the sidebar. Similarly, if a ticket's information hadn't changed in quite a while, it could suggest that the ticket be deleted.

However, for now, we believe that our experience with Sideshow provides two lessons for the research community. First, users are willing to give up a portion of their screen space for a peripheral awareness application. Second, spending time to make sure peripheral awareness applications focus on information that's important to users is critical for success. Peripheral awareness displays should be able to be personalized in such a way that they help users

stay aware of information that's critical to them, and that will often mean building a system that goes beyond watching stock prices, news stories, and weather forecasts.

ACKNOWLEDGMENTS

We're very appreciative of Mike Boyle for developing the first Visual Basic prototype of Sideshow while he was an intern with us during the summer of 2000. We're also extremely grateful for the hundreds of people throughout our company who ran Sideshow and sent us bug reports, design suggestions, and words of encouragement.

REFERENCES

1. Atkins, D., Boyer, D., Handel, M., Herbsleb, J., Mockus, A., Wills, G. The Product Development Collaboratory at Lucent Technologies. http://www.bell-labs.com/org/11359/colab_prod/
2. Cutrell, E., Czerwinski, M., and Horvitz, E. (2001). Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance. *Proceedings of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 2001).
3. Dourish, P., and Bellotti, V. (1992). Awareness and Coordination in Shared Workspaces. *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (CSCW 1992).
4. Dourish, P., and Bly, S. (1992). Portholes: Supporting Awareness in a Distributed Work Group. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 1992).
5. Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T., and Segall, B. (1999). Instrumenting and Augmenting the Workaday World with a Generic Notification Service Called Elvin. *Proceedings of 6th European Conference on Computer Supported Cooperative Work* (ECSCW 1999).
6. Greenberg, S., and Rounding, M. (2001). The Notification Collage: Posting Information to Public and Personal Displays. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2001).
7. Grudin, J. (2001). Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2001).
8. Harrison, B., Ishii, H., Vicente, K., and Buxton, W. (1995). Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 1995).
9. Heiner, J., Hudson, S., and Tanaka, K. (1999). The Information Percolator: Ambient Information Display in a Decorative Object. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST 1999).
10. Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-Sensitive Alerting. *Proceedings of the Conference on Uncertainty and Artificial Intelligence* (UAI 1999).
11. Ishii, H., and Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 1997).
12. McFarlane, D. (1999). Coordinating the Interruption of People in Human-Computer Interaction. *Proceedings of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 1999).
13. MacLean, A., Carter, K., Lovstrand, L, and Moran, T. (1990). User-Tailorable Systems: Pressing the Issues with Buttons. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 1990).
14. McCrickard, D., Catrambone, R., and Stasko, J. (2001). Evaluating Animation in the Periphery as a Mechanism for Maintaining Awareness. *Proceedings of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 2001).
15. Maglio, P., and Campbell, C. (2000). Tradeoffs in Displaying Peripheral Information. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2000).
16. McCrickard, D. (1999). Maintaining Information Awareness with Irwin. *Proceedings of the World Conference on Educational Multimedia/Hypermedia and Educational Telecommunications* (ED-MEDIA 1999).
17. Miller, T., and Stasko, J. (2001). The InfoCanvas: Information Conveyance through Personalized, Expressive Art. *Extended Abstracts from the ACM Conference on Human Factors in Computing Systems* (CHI 2001).
18. Pacey, M., and MacGregor, C. (2001). Auditory Cues for Monitoring a Background Process: A Comparative Evaluation. *Proceedings of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 2001).
19. Pedersen, E., and Sokoler, T. (1997). AROMA: Abstract Representation of Presence Supporting Mutual Awareness. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 1997).
20. Tang, J., Yankelovich, N., Begole, J., Van Kleek, M., Li, F., and Bhalodia, J. (2001). ConNexus to Awarenex: Extending Awareness to Mobile Users. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2001).
21. Tollmar, K., Sandor, O., and Schomer, A. (1996). Supporting Social Awareness @ Work, Design, and Experience. *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (CSCW 1996).
22. Weiser, M., and Brown, J. (1996). Designing Calm Technology. *PowerGrid Journal*, v1.01, July 1996. (see http://nano.xerox.com/hypertext/weiser/calmtech/ calmtech.htm