CybreMinder: A Context-Aware System for Supporting Reminders

Anind K. Dey and Gregory D. Abowd

Future Computing Environments Group
College of Computing and GVU Center
Georgia Institute of Technology, Atlanta, GA, USA 30332-0280
{anind, abowd}@cc.gatech.edu

Abstract. Current tools do not provide adequate support to users for handling reminders. The main reason for this is the lack of use of rich context that specifies when a reminder should be presented to its recipient. We describe CybreMinder, a prototype context-aware tool that supports users in sending and receiving reminders that can be associated to richly described situations involving time, place and more sophisticated pieces of context. These situations better define when reminders should be delivered, enhancing our ability to deal with them more effectively. We describe how the tool is used and how it was developed using our previously developed Context Toolkit infrastructure for context-aware computing.

1 Introduction

A reminder is a special type of message that we send to ourselves or others, to inform us about some future activity that we should engage in. For example, a colleague might send us a reminder asking us to bring a copy of a paper to our next meeting.

We use reminders to signal others and ourselves that a task still exists to be worked on and/or that a task is ready for further processing We use reminders to re-establish needed information in short-term memory so that the trigger conditions for these reminders can be satisfied [14].

Reminders have two main features — a signal and a description. The signal is used to indicate something is to be remembered. An example of an audio-based signal is an alarm on an alarm clock. Lights flashing in a theatre or a note pinned to a door are examples of visual signals. The description is used to explain what needs to be remembered. This can vary from being non-descriptive, in the case of the alarm clock, to being partially descriptive, in the case of a icon which provides only a few cues as to what needs remembering, to being fully descriptive, in the case of an e-mail message or handwritten note that provides all relevant details of the reminder.

We currently have a number of tools and strategies at our disposal to help us keep track of reminders. However, studies have shown that users still have difficulty dealing with reminders [6]. Difficulties stem from a number of issues regarding the use of signals. Current reminder systems, acting as a form of externalized memory, do not

P. Thomas and H.-W. Gellersen (Eds.): HUC 2000, LNCS 1927, pp. 172-186, 2000.

present appropriate signals at appropriate times. More specifically, these tools are not sufficient because they are not proactive and do not make use of rich contextual information to trigger reminders at appropriate times in appropriate locations. Herstad *et al.* claim that in order to build useful, functional and powerful tools for supporting human-human interaction, we must take context into account [9]. For example, to be most effective, a reminder to bring a paper to a meeting should be delivered when we are leaving our office and heading towards the meeting, and not when we happen to read our e-mail. We will investigate this idea further by looking at traditional ways of handling reminders, indicating how insufficient use of context causes problems.

By reviewing existing reminder tools, we will show that users have trouble dealing with reminders due to the lack of use of rich context. We will then propose a list of features that an ideal reminder should support. We describe CybreMinder, a reminder tool that supports these features and some scenarios that it currently supports. Finally, we describe CybreMinder's system architecture and how it leverages off an existing context-sensing infrastructure (the Context Toolkit [4,16]) to allow the specification of situations in which reminders can be delivered.

2 Current Reminder Tools

We use a variety of tools that help us in creating and managing our reminders. In this section, we examine these tools and investigate our claim that they do not use enough context information to adequately support our needs. This brief review of reminder tools will lead us to a list of desirable features for a context-aware reminder system.

2.1 Paper To-Do Lists

A common reminder tool is a to-do list written on a piece of paper. The to-do list may contain both traditional calendar/scheduler information and a set of tasks that need to be completed. While it is simple to create a list, it is not so easy to remember to use it in the appropriate situation. A to-do list lacks the ability to proactively remind us when an item on the list needs to be accomplished. Instead, the list creator must remember to check it often, to determine which items need/can be accomplished. In other words, a to-do list provides reminders with descriptions, but no signals.

2.2 E-Mail Mailbox

An e-mail mailbox is often used as an informal to-do list. Some people send themselves e-mail as a reminder to perform some activity at a later date. A study of e-mail tool usage showed that when checking their e-mail, people often flag messages containing to-do items to create a visual reminder [8]. Another strategy is to file them in a special mail folder, creating the electronic equivalent of a paper-based to-do list. As with the paper-based to-do list, e-mail tools cannot proactively remind us of to-do items. We are forced to repeatedly review these flagged or stored messages, in an

attempt to ascertain which to-do items can be handled at the time. Again, this is an example of a reminder with descriptions, but no signals. One advantage of e-mail over paper is that people can use e-mail to create and send reminders to others. However, it suffers from the disadvantage of not being as readily available as paper.

2.3 Post-It Notes

Another common strategy is to use post-it notes, paper or virtual [2,15], placed in locations where the intended recipients can view them. The visibility of post-it notes in the environment provides a signal to recipients that something needs to be remembered. The content of the note provides the description of what is to be remembered. Because post-it notes are always visible to the intended recipients, there is no way to determine when they are valid. For example, a reminder to call someone may only be valid before 10 p.m. but the reminder is still visible after 10 p.m., unless it is explicitly removed. The paper post-it notes also have the disadvantage that they can also be viewed by anyone, not just the intended recipients. This is another example of a reminding tool with inappropriate signaling capabilities. Post-it notes can only provide a signal based on their location in the environment, indicating that they are only useful for location-based reminders. An additional disadvantage of post-it notes is that they do not support the ability to collect all reminders in a single artifact, unlike email folders or to-do lists.

2.4 Personal Information Management Tools

Personal information management (PIM) tools such as electronic calendar and to-do list programs suffer from a similar problem as post-it notes. While post-it notes are useful for reminders where location is the only useful context, these PIM tools only have affordances for temporal context: a meeting is at a certain time or a task must be completed by a certain date. Current PIM tools can only provide a signal based on the current time, making them no more intelligent than a simple alarm clock. The tool provides an audio or vibration signal and it is our responsibility to retrieve the description or content of the reminder. Both types of cues may be inappropriate in various situations (e.g. audio cues in a meeting are disruptive and vibration cues while jogging may not be noticed). This suggests that the manner in which reminders are delivered is also extremely important.

2.5 Human Assistant

Another “tool” used to manage reminders is a human assistant. We often rely on a personal assistant (secretary, spouse, etc.) to remind us of scheduled events and tasks that require attention. The assistant acts as a mediator between the actual set of things to be remembered and ourselves, creating reminders from a variety of communication media (phone messages, faxes, e-mail, etc.), presenting them in the appropriate situation and using an appropriate delivery mechanism. However, even a personal assistant

may not be enough. In most cases, personal assistants can only present reminders when they are co-located with us. They can provide two types of reminders, those that are relevant right now and those that are relevant in a future context when they will not be with us. It is the second case that is troublesome. How many of us have been reminded to pick up an item from a grocery store as we are leaving for work, only to forget to stop at the store on the way home? An ideal assistant would provide the reminder in the context that maximizes the chance for appropriate action.

2.6 Desired Features of a Reminder System

The externalized memory tools we have discussed are simple to use, but all take limited advantage of context for signaling; or for indicating that a reminder is relevant in our current *situation*. Human assistants come closest to being the ideal reminder tool, but as shown, there are opportunities to improve on their capabilities, and we cannot all have human assistants. Based on our analysis of current tools, here is a list of the features an ideal reminder tool should support:

- the use of rich context for specifying reminders, beyond simple time and location and for proactively determining when to deliver them;
- the ability for users and third parties to submit reminders;
- the ability to create reminders using a variety of input devices;
- the ability to receive reminders using a variety of devices, appropriate to the user's situation;
- the use of reminders that include both a signal that something is to be remembered and a full description of what is to be remembered; and
- allowing users to view a list of all active reminders.

3 Related Work

There has not been a lot of previous work in the area of context-aware reminders. As discussed in the previous section, commercial efforts have focused on e-mail tools that use no context or PIM tools that use only time. Another system that uses time-aware reminders is Lifestreams [6]. Lifestreams is a system for organizing documents that is intended to replace conventional files and directory structures. Instead, Lifestreams organizes documents temporally, based on when they were created, received, and/or modified. The beginning of a stream contains the oldest documents while the end of a stream contains the most recently created documents. The interface allows users to even visit the future portion of the document stream. When a user creates a document in the future portion of the stream, they are effectively creating a time-based reminder. When they return to present time, these documents are hidden and only appear when present time matches the future time of the documents.

The comMotion project [13] moved beyond this by using a combination of location and time information to deliver relevant messages. When a reminder message is created, a location is associated with it. Then, when the intended recipient arrives at that

location (work or a grocery store, for example), the messages associated with that location are delivered via speech synthesis. In addition, when a user arrives at work, her calendar events for that day are delivered, taking advantage of time as well as location information.

Proem is a wearable computer-based system that supports profile-based cooperation [11]. Wearers can write simple rules that indicate their interests in other people. When someone physically close to the wearer has a profile that matches one or more of his interests, Proem can alert him. Interests are limited to fairly static pieces of information such as names and personal interests and hobbies.

Memory Glasses is a wearable computer-based context-aware reminder system [3]. It proposes the use of time, location, and activity to deliver reminders. It focuses on personal context and uses body-worn sensors (a camera and a microphone) to determine what activity the wearer is engaged in, including walking down stairs or taking part in a conversation. When Memory Glasses determines the current activity, reminders associated with that activity are presented to the using audio output. Memory Glasses proposes that knowledge of this activity may be used to better determine when it is appropriate to interrupt the wearer with a reminder.

While these systems address many of the features of an ideal reminder tool, they are limited by their restricted use of context. The notion of context is quite rich and encompasses many information types beyond location, time and activity, such as identity, physical/environmental conditions, as well as information about other individuals besides the user [5,18]. As the context associated with a reminder is made richer, the system's ability to deliver the reminder in the appropriate situation is improved. The CybreMinder reminder tool we will present in the next section attempts to address all the features of the ideal reminder tool, concentrating on increasing the variety of context used to associate with reminders. It is not intended to replace existing calendar or to-do list tools, but to augment them.

4 The CybreMinder Tool

To aid our investigation of reminder tools and interfaces, we built the Java-based CybreMinder tool. It has two main parts — reminder creation and reminder delivery.

4.1 Reminder Creation

When users launch CybreMinder, they are presented with an interface that looks quite similar to an e-mail creation tool. As shown in Figure 1, users can enter the names of the recipients for the reminder. The recipients could just be themselves, indicating a personal reminder, or a list of other people, indicating a third party reminder is being created. The reminder has a subject, a priority level (ranging from lowest to highest), a body in which the reminder description is placed, and an expiration date. The expiration date indicates the date and time at which the reminder should expire and be delivered, if it has not already been delivered.

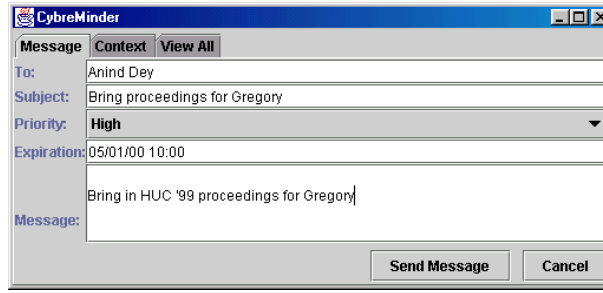

Fig. 1. CybreMinder reminder creation tool

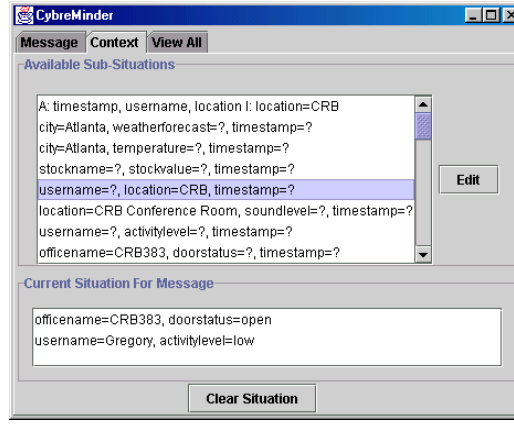

Fig. 2. CybreMinder situation editor

In addition to this traditional messaging interface, users can select the Situation tab and be presented with the situation editor (Figure 2). This interface allows dynamic construction of an arbitrarily rich situation, or context that is associated with the reminder being created. The interface consists of two main pieces for creating and viewing the situation. Creation is assisted by a dynamically generated list of valid sub-situations that are currently supported by the CybreMinder infrastructure (as assisted by the Context Toolkit described later). When the user selects a sub-situation, they can edit it to fit their particular situation. Each sub-situation consists of a number of context types and values. For example, in Figure 2, the user has just selected the sub-situation that a particular user is present in the CRB building at a particular time. The context types are the user's name, the location (set to CRB) and a timestamp.

In Figure 3, the user is requiring the user name to be "Anind Dey", and is not using time. This sub-situation will be satisfied when Anind Dey is in the location 'CRB'.

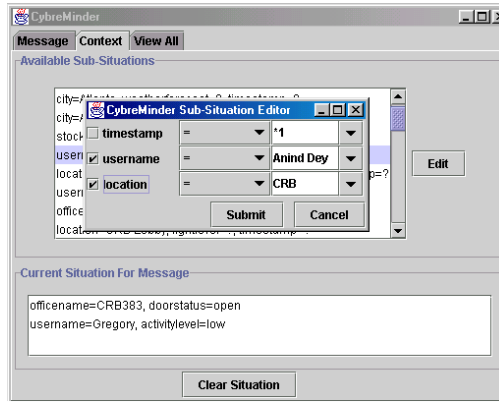

Fig. 3. Sub-situation editor

The user indicates which context types are important by selecting the checkbox next to those attributes. For the types that they have selected, users may enter a relation other than '='. For example, the user can set the timestamp after 9 p.m. by using the '>' relation. Other supported relations are '>=', '<', and '<='. For the context value, users can either choose from a list of pre-generated values, or enter their own.

At the bottom of Figure 2, the currently specified situation is visible. The overall situation being defined is the conjunction of the sub-situations listed. Once a reminder and an associated situation have been created, the user can send the reminder. If there is no situation attached, the reminder is delivered immediately after the user sends the reminder. However, unlike e-mail messages, sending a reminder does not necessarily imply immediate delivery. If a situation is attached, the reminder is delivered to recipients at a future time when all the sub-situations can be simultaneously satisfied. If the situation cannot be satisfied before the reminder expires, the reminder is delivered both to the sender and recipients with a note indicating that the reminder has expired.

4.2 Reminder Delivery

Thus far, we have concentrated on the process of creating context-aware reminders. We will now describe the delivery process. When a reminder can be delivered, either because its associated situation was satisfied or because it has expired, CybreMinder determines what is the most appropriate delivery mechanism for each reminder recipient. The default signal is to show the reminder on the closest available display, augmented with an audio cue. However, if a recipient wishes, they can specify a configuration file that will override this default.

A user's configuration file contains information about all of the available methods for contacting the user, as well as rules defined by the user on which method to use in which situation. If the recipient's current context and reminder information (sender identity and/or priority) matches any situation defined in his configuration file, the specified delivery mechanism is used. Currently, we support the delivery of reminders via SMS on a mobile phone, e-mail, displaying on a nearby display (wearable, handheld, or static CRT) and printing to a local printer (to emulate paper to-do lists).

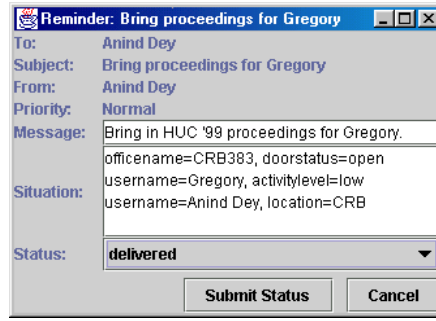

Fig. 4. Delivered reminder

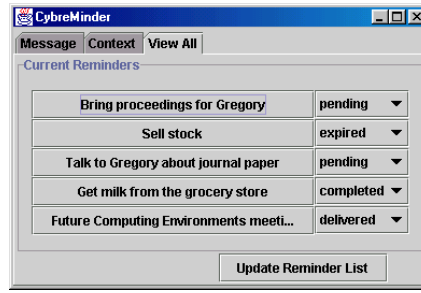

Fig. 5. List of all reminders

For the latter three mechanisms, both the reminder and associated situation are delivered to the user. Delivery of the situation provides additional useful information to the user, helping them understand why the reminder is being sent at this particular time. Along with the reminder and situation, users are given the ability to change the status of the reminder (Figure 4). A status of “completed” indicates that the reminder has been addressed and can be dismissed. The “delivered” status means the reminder has been delivered but still needs to be addressed. A “pending” status means that the reminder should be delivered again when the associated situation is next satisfied. Users can explicitly set the status through a hyperlink in an e-mail reminder or through the interface shown in Figure 4.

Since SMS messages have limited length, only the subject of a reminder is delivered when using this delivery mechanism. Users receiving such an SMS message have the option of going to a networked device and launching their interface (Figure 1) to CybreMinder. By selecting the View All tab, users can view a personalized list of all reminders and can change the status of any of these reminders (Figure 5).

5 Example Reminders

In this section, we describe a range of reminders and situations that users can create using CybreMinder, moving from simple situations towards more complex situations. The situations are only limited by the context that can be sensed. Table 1 gives the natural language and CybreMinder descriptions of the illustrated situations below.

Table 1. Natural language and CybreMinder descriptions of scenarios in Section 5

Situation	Natural Language Description	CybreMinder Description
Time	9:45 am	Expiration field: 9:45 am
Location	Forecast is for rain and Bob is leaving home	City = Atlanta, WeatherForecast = rain Username = Bob, Location = Bob's front door
Co-Location	Sally and colleague are co-located	Username = Sally, Location = *1 Username = Bob, Location = *1
Complex #1	Stock price of X is over $50, Bob is alone and has free time	StockName = X, StockPrice > 50 Username = Bob, Location = *1 Location = *1, OccupantSize = 1 Username = Bob, FreeTime > 30
Complex #2	Sally is in her office has some free time, and her friend is not busy	Username = Sally, Location = Sally's office Username = Sally, FreeTime = 60 Username = Tom, ActivityLevel = low

5.1 Time-Based Reminder

Like many of the other systems previously described, CybreMinder allows reminders to be triggered based on a simple time context. In this scenario, Sally has a meeting at 10 a.m. tomorrow. She wants to send a reminder to herself fifteen minutes before the meeting occurs, so that she has time to walk to the meeting. She can simply set the expiry date to be tomorrow's date and 9:45 a.m.

5.2 Location-Based Reminder

In this scenario, Bob wants to remind himself to take his umbrella to work because it is supposed to rain this afternoon. He keeps the umbrella near his apartment door, so he wants to receive the reminder as he approaches the door. Here, he can simply create a situation with only one sub-situation: he is at his front door. In CybreMinder terms, he sets the username to his name and location to his front door. This situation can be made slightly more complex. If Bob is sending the reminder the night before, then he may want to add a time attribute and set it to be greater than 7:00 a.m. By doing so, the reminder will not be triggered and displayed each time he leaves his apartment that night. It will only be displayed when he approaches the door after 7:00 a.m. the next morning. Pushing on this scenario a little more, Bob does have to know ahead of time that it is going to rain. He can simply create a reminder that is to be delivered whenever the forecast calls for rain and he is leaving his apartment.

5.3 Co-location-Based Reminder

Of the systems we reviewed, only Proem [11] supported proactive reminders when two or more people were co-located in an arbitrary location. It can be argued that post-it notes could be used in this setting, although it currently breaks normal social conventions to stick post-it notes to people. An example co-location scenario follows: Sally wants to engage a colleague in a discussion about an interesting paper she read, but forgets when she sees her colleague. She can create a context-aware reminder that will be delivered when she is in close proximity with her colleague. The situation she creates is slightly more complex than the ones we have discussed so far, and it makes use of variables. Variables allow users to create relationships between sub-situations. First Sally creates an initial sub-situation where she sets the user name to be her colleague's name and the location to be variable (indicated in Table 1 by *1). Then, she creates a second sub-situation, where she sets the user name to be her name and the location to the variable used in the first sub-situation. Now when Sally and her colleague are in the same arbitrary location, the reminder will be delivered.

5.4 Complex Reminder

CybreMinder supports the unlimited use of rich context, allowing users to create as rich a situation as can be sensed. We describe two such situations. In the first scenario, Bob owns stock in Company X and has decided to sell that stock when it is valued over $50 per share. He only wants to be reminded to sell, however, when he is alone and has free time. To create this situation to signal a reminder to sell, Bob creates a number of sub-situations: stock price of company X > $50, Bob is the only occupant of his location, and Bob's schedule shows that he has > 30 minutes before his next meeting. When this situation occurs, Bob receives the reminder to sell his stock.

In our second complex scenario, Sally needs to make a phone call to her friend Tom. She wants to receive a context-aware reminder when she arrives at her office, has some free time in her schedule, and her friend is not busy. To create this situation, she creates three sub-situations: Sally is in her office, Tom's activity status is low, and Sally has at least one hour before her next appointment.

6 The CybreMinder Architecture

In the previous two sections, we described how CybreMinder works from the user's perspective. Here, we discuss how CybreMinder was built. When users write situations to be associated with reminders, CybreMinder must have a way to determine when they have been realized. It uses the Context Toolkit[1] for this purpose.

[1] The Context Toolkit and tutorial can be downloaded from http://www.cc.gatech.edu/fce/contexttoolkit.

6.1 The Context Toolkit

The Context Toolkit is a software toolkit that aids in the building of context-aware applications [4,16]. It promotes three main concepts for building context-aware applications; separation of context sensing, or acquisition, from context use; context aggregation; and context interpretation. It relieves developers from having to deal with how to sense and access context information, allowing them instead to concentrate on how to use the context. It provides simplifying abstractions like aggregation and interpretation to make it easier for applications to obtain the context they require. Aggregation provides "one-stop shopping" for context about an entity, allowing application designers to think in terms of high level information, rather than low-level details.

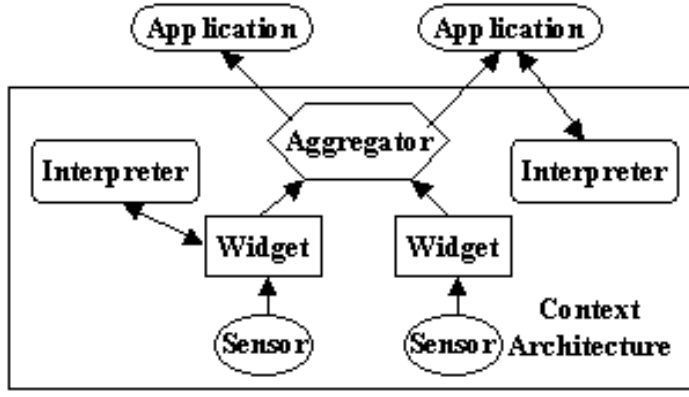

Fig. 6. Context Toolkit components: arrows indicate data flow

The architecture makes it easy to add the use of context to existing applications that don't use context and to evolve applications that already use context. In addition, the architecture makes context-aware applications resistant to changes in the context-sensing layer. It encapsulates changes and the impact of changes, so applications do not need to be modified.

The Context Toolkit consists of three basic building blocks: context widgets, context aggregators and context interpreters. Figure 6 shows the relationship between the context components and applications. Context widgets encapsulate information about a single piece of context, such as location or activity, for example. They provide a uniform interface to components or applications that use the context, hiding the details of the underlying context-sensing mechanism(s). They allow other components to both poll and subscribe to the context information they maintain. Widgets are mainly responsible for collecting information about the environment. However, they also support services that allow them to affect the environment. For example, a Light widget that detects the intensity of the light in a particular location, and have a service that controls a lamp to change the intensity.

A context aggregator is very similar to a widget, in that it supports the same set of features as a widget. The difference is that an aggregator aggregates multiple pieces of context. In fact, it is responsible for the entire context about a particular entity (person,

place, or object). Aggregation facilitates the access of context by applications that are interested in multiple pieces of context about a single entity. A context interpreter is used to abstract or interpret context. For example, a context widget may provide location context in the form of latitude and longitude, but an application may require the location in the form of a street name. A context interpreter may be used to provide this abstraction.

Context components are intended to be persistent, running 24 hours a day, 7 days a week. They are instantiated and executed independently of each other in separate threads and on separate computing devices. The Context Toolkit makes the distribution of the context architecture transparent to context-aware applications, mediating all communications between applications and components. A discovery protocol allows components to communicate with each other without knowing about the existence of each other at compile or instantiation time.

6.2 CybreMinder and the Context Toolkit

When CybreMinder launches, it uses the Context Toolkit discovery protocol to detect what context components are currently available. It analyzes this and determines what sub-situations are available for a user to work with. The sub-situations are simply the collection of subscription callbacks that the context widgets and context aggregators provide. For example, an IdentityPresence context widget contains information about the presence of individuals in a particular location (specified at instantiation time). The callback it provides has three attributes: a user name, a location, and a timestamp. The location is a constant, set to “home”, for example. The constants in each callback are used to populate the menus from which users can select values for attributes.

When the user creates a reminder with an associated situation, the reminder is sent to the aggregator responsible for maintaining context about the recipient. CybreMinder can be shut down any time after the reminder has been sent to the recipient’s aggregator. The recipient’s aggregator is the logical place to store all reminder information intended for the recipient because it knows more about the recipient than any other component and is always available. This aggregator analyzes the given situation and creates subscriptions to the necessary aggregators and widgets so that it can determine when the situation has occurred. It also creates a timer thread that awakens when the reminder is set to expire. Whenever the aggregator receives a subscription callback, it updates the status of the situation in question. When all sub-situations are satisfied, the entire situation is satisfied, and the reminder can be delivered.

The recipient’s aggregator contains the most up-to-date information about the recipient. It tries to match this context information along with the reminder sender and priority level with the rules defined in the recipient’s configuration file. The recipient’s context and the rules consist of collections of simple attribute name-value pairs, making them easy to compare. When a delivery mechanism has been chosen, the aggregator calls a widget service that can deliver the reminder appropriately. For example, a display widget provides information about the display capabilities of a device. It also provides a service that allows other components to display information on that device. Similarly, e-mail and SMS services exist in the Context Toolkit.

Services can also return information to the component that calls them. For example, the display service not only shows the reminder and associated situation, but also a form allowing the user to set the state of this reminder. The user input to this form is sent back to the recipient's aggregator, which can update the reminder status. In the case of SMS, when the user must set the status using the CybreMinder, the application contacts the user's aggregator and queries for all the reminders and associated information. The application sends any updated status information to the aggregator.

7 Conclusions and Future Work

The goal of CybreMinder is to provide users with a tool that provides appropriate support for dealing with reminders. In particular, our objective is to support all the features of an ideal reminder tool:

- use of rich context for specifying reminders, beyond simple time and location and for proactively determining when to deliver them;
- ability for users and third parties to submit reminders;
- ability to create reminders using a variety of input devices;
- ability to receive reminders using a variety of devices, appropriate to the user's situation;
- use of reminders that include both a signal that something is to be remembered and a full description of what is to be remembered; and
- allowing users to view a list of all active reminders.

We believe that we have been mostly successful in this objective. We provide some support for all of these features, except for the ability to create reminders using a variety of input devices. We will discuss each of these features in turn.

The first feature, allowing for the use of rich context in reminders, is the most important feature for a reminder tool and is the one that is most lacking in existing reminder tools. By leveraging off of the Context Toolkit's ability to acquire and distribute context, we allow users to create arbitrarily complex situations to attach to reminders and to create custom rules for governing how reminders should be delivered to them. Users are not required to use templates or hardcoded situations, but can use any context that can be sensed and is available from their environment. From initial use of the system, we have found that while the interface supports the specification of complex situations, it can be complex to use, particularly when variables are involved. We would like to find the correct balance of sophistication and simplicity in an effort to improve the interface. One potential solution is a suite of special-purpose reminders with highly simplified interfaces that suit their specific use.

By using a discovery protocol for determining what context is available, we attempt to limit users in creating situations that CybreMinder is able to detect. However, it is possible for users to create situations that CybreMinder cannot detect. Some of these situations can be caught by the aggregator, but not all. We intend to improve the checking ability of CybreMinder.

Part of the reminder creation process is the specification of recipients. A user can set the recipients of the reminder to be herself, herself and others, or just others. In this

way, CybreMinder supports the ability to send reminders to yourself or third parties. Currently, users can only create reminders using the Java-based CybreMinder. This application can run on any networked device that can support Java, including desktop computers, WinCE devices and wearable computers. However, the Context Toolkit does not require that applications be written in Java. We envision, in the near future, creating simplified versions of CybreMinder that can be executed on a Palm Pilot, a pager, or a mobile phone, which a user interacts with not only using text, but also pen and speech input. We would also like to support the automatic creation of reminders from user's calendars and to-do lists.

On the delivery side, CybreMinder sends both the reminder and the associated situation to a service for display (via e-mail, available screen, or SMS). The quality of the reminder signal and the completeness of the reminder description depend on the service being used. E-mail provides a poor signal if the user is not at reading their e-mail at the reminder time, but does present a complete description. Displaying a reminder on a nearby screen with an audio cue provides both a good signal and a complete description. SMS provides a very good signal but only a partial description. Likewise, there are advantages and disadvantages to automatically printed reminders.

By supporting a greater variety of devices and display services, we can allow users to make better personal choices about how they want to receive reminders (both in terms of the signal and description) in various situations. CybreMinder delivers a reminder when the associated situation has been realized, and chooses the delivery mechanism/service based on the recipient's current context. However, it does not take into account how interruptible the recipient is. We realize that the use of static user configuration files is not the answer, but determining interruptiblity is an enormous unsolved research problem [10,14,17,19], and as researchers make progress in this area, we would like to improve CybreMinder accordingly.

Initial responses to CybreMinder have been promising. We intend to expand our current user population and perform an objective evaluation of CybreMinder and comparison to existing reminder tools. We would also like to examine the use of the CybreMinder reminder tool as part of a larger context-aware messaging system.

8 Acknowledgements

We would like to thank the Future Computing Environments research group for contributing to these ideas. This work was supported in part by a NSF CAREER Grant # 9703384, a Motorola University Partnerships in Research grant and the ITO division of DARPA through the Expeditions/Ubiquitous Computing program.

References

1. Bergqvist, J., Ljungberg, F.: ComCenter: A Person Oriented Approach to Mobile Communication. Extended abstract In *Proceedings of CHI 2000* (2000) 123–124

2. Brown, P.J.: The Stick-e Document: A Framework for Creating Context-Aware Applications. Electronic Publishing (1996) 259–272
3. DeVaul, R.W., Clarkson, B., Pentland, A.: The Memory Glasses: Towards a Wearable Context Aware, Situation-appropriate Reminder System. In *CHI 2000 Workshop on Situated Interaction in Ubiquitous Computing* (2000)
4. Dey, A.K., Abowd, G.D., Salber, D.: A Context-Based Infrastructure for Smart Environments. *1st International Workshop on Managing Interactions in Smart Environments (MANSE'99)* (1999) 114–128
5. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. In *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness* (2000)
6. Fertig, S., Freeman, E., Gelernter, D.: "Finding and Reminding" Reconsidered. SIGCHI Bulletin, Vol. 28 (1996)
7. Gellersen, H-W.: EMC: Environment-Mediated Communication. *International Workshop on Interactive Applications of Mobile Computing (IMC'98*) (1998)
8. Gwizdka, J.: Timely Reminders: A Case Study of Temporal Guidance in PIM and Email Tools Usage. Extended abstract in *Proceedings of CHI 2000* (2000) 163–164
9. Herstad, J. Van Thanh, D., Audestad, J.A.: Human-Human Communication in Context. *International Workshop on Interactive Applications of Mobile Computing IMC'98* (1998)
10. Horvitz, E.: Mixed-Initiative User Interfaces. In *Proceedings of CHI 99* (1999) 159–166
11. Korteum, G., Segall, Z., Thompson, T.G.C.: Close Encounters: Supporting Mobile Collaboration through Interchange of User Profiles. In *Proceedings of HUC'99* (1999) 171–185
12. Ljungstrand, P.: Context-awareness in distributed communication systems. In *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness* (2000)
13. Marmasse, N.: comMotion. Extended abstract in *Proceedings of CHI'99* (1999) 320–321
14. Miyata, Y., Norman, D.A.: Psychological Issues in Support of Multiple Activities. User Centered Design, edited by Norman, D.A., Draper, S.W. Chapter 13 (1986) 265–284
15. Rekimoto, J., Ayatsuka, Y., Hayashi, K.: Augment-able Reality: Situated Communication through Physical and Digital Spaces. In *Proceedings of 2nd International Symposium on Wearable Computers (ISWC'98)* (1998) 68–75
16. Salber, D., Dey, A.K., Abowd, G.D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of CHI'99* (1999) 434–441
17. Sawhney, N., Schmandt, C.: Nomadic Radio: Scaleable and Contextual Notification for Wearable Audio Messaging. In *Proceedings of CHI'99* (1999) 96–103
18. Schmidt, A., Beigl, M. Gellersen, H-W. There is More to Context than Location: Environment Sensing Technologies for Adaptive Mobile User Interfaces. *Workshop on Interactive Applications of Mobile Computing IMC'98* (1998)
19. Stringer, M., Eldridge, M, Lamming, M.: Towards a Deeper Understanding of Task Interruption. In *CHI Workshop on Situated Interaction in Ubiquitous Computing* (2000)