Context-Aware Computing Applications

Bill N. Schilit, Norman Adams, and Roy Want

Abstract

This paper describes software that examines and reacts to an individual's changing context. Such software can promote and mediate people's interactions with devices, computers, and other people, and it can help navigate unfamiliar places. We believe that a limited amount of information covering a person's proximate environment is most important for this form of computing since the interesting part of the world around us is what we can see, hear, and touch. In this paper we define context-aware computing, and describe four categories of context-aware applications: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions. Instances of these application types have been prototyped on the PARCTAB, a wireless, palm-sized computer.

1 Introduction

Our investigation focuses on an extended form of mobile computing in which users employ many different mobile, stationary and embedded computers over the course of the day. In this model computation does not occur at a single location in a single context, as in desktop computing, but rather spans a multitude of situations and locations covering the office, meeting room, home, airport, hotel, classroom, market, bus, etc. Users might access their computing resources from wireless portable machines and also through stationary devices and computers connected to local area networks.

We call this collection of mobile and stationary computing devices that are communicating and cooperating on the user's behalf a *mobile distributed computing system*. This form of computing is broader than mobile computing because it concerns mobile *people* not just mobile *computers*. These systems aim to provide ubiquitous access to information, communication, and computation.

One significant aspect of this emerging mode of computing is the constantly changing execution environment. The processors available for a task, the devices accessible for user input and display, the network capacity, connectivity, and costs may all change over time and place. In short, the hardware configuration is continually changing. Similarly, the computer user may move from one location to another, joining and leaving groups of people, and frequently interacting with computers while in changing social situations.

2 Context-Aware Computing

One challenge of mobile distributed computing is to exploit the changing environment with a new class of applications that are aware of the context in which they are run. Such *context-aware software* adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment.

Three important aspects of context are: where you are, who you are with, and what resources are nearby (see Figure 2.). Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation; e.g., whether you are with your manager or with a co-worker.

We are investigating these kinds of applications using the PARCTAB [1, 7], a small hand held device which uses an infrared-based cellular network for communication. The tab acts as a graphics terminal, and most applications run on remote hosts. This design exploits remote processing power to achieve a smaller and cheaper device. For input, the

[1] This work was supported by Xerox. Portions were also supported by ARPA under contract DABT63-91-C-0027.

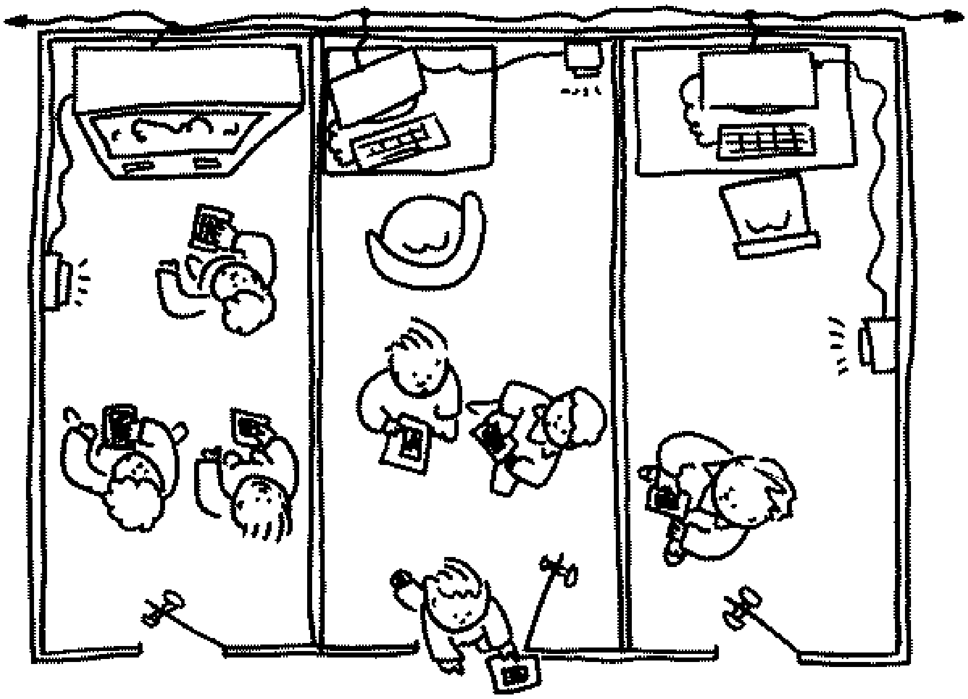

Figure 1: A Context-Aware Computing System (PARCTAB)

tab has three finger-operated buttons on the grip, and a touch sensitive screen. For output, the tab has a 128x64 pixel display and a piezo-electric speaker. When wired with an infrared transceiver, a room becomes a cell in the infrared network. The bandwidth available for all tabs in a cell is 19,200 bps.

The combination of room-sized cells, and tabs that periodically send an identifying packet (i.e., beacon), permits accurate location monitoring even when the device is not being used. The system notifies applications of location changes, and also provides location information to a public service that collects and redistributes information about objects and their locations [9]. Other systems might learn about location by other means: for example, by using global positioning (GPS) or dead-reckoning, or simply by monitoring workstation interactions. An alternative to the tab system's locating scheme – where the mobile devices beacon – is a scheme where stationary devices broadcast a message identifying their fixed location. Mobile computers can listen to these broadcasts to determine their own locations.

The devices and locating strategy described above combine to form a mechanism for building context-aware applications. In the following sections we describe four categories of context-aware applications. These categories are the product of two points along two orthogonal dimensions (see Table 2.): whether the task at hand is getting information or doing a command, and whether it is effected manually or automatically.

	manual	automatic
information	proximate selection & contextual information	automatic contextual reconfiguration
command	contextual commands	context-triggered actions

Table 1: Context-Aware Software Dimensions

Name	Room	Distance
caps	35-2200	200ft
claudia	35-2108	30ft
perfector	35-2301	20ft
snoball	35-2103	100ft

(a)

Distance	Name	Room
20ft	perfector	35-2301
30ft	claudia	35-2108
100ft	snoball	35-2103
200ft	caps	35-2200

(b)

Name	Room	Distance
caps	35-2200	200ft
claudia	**35-2108**	**30ft**
perfector	**35-2301**	**20ft**
snoball	35-2103	100ft

(c)

Name	Room	Distance
caps	35-2200	200ft
claudia	35-2108	30ft
perfector	35-2301	20ft
snoball	35-2103	100ft

(d)

Table 2: UI Techniques for Proximate Selection

2.1 Proximate Selection

Proximate selection is a user interface technique where the located-objects that are nearby are emphasized or otherwise made easier to choose. In general, proximate selection involves entering two variables, the “locus” and the “selection.” However, of particular interest are user interfaces that automatically default the locus to the user’s current location.

There are at least three kinds of located-objects that are interesting to select using this technique. The first kind is computer input and output devices that require co-location for use. This includes printers, displays, speakers, facsimiles, video cameras, thermostats, and so on. The second kind is the set of objects that you are already interacting with, and which need to be addressed by a software process. This includes people in the same room to whom you would like to “beam” a document. The third kind is the set of places one wants to find out about: restaurants, night clubs, gas stations, and stores, or more generically, exits and entrances. Consider an electronic yellow pages directory that, instead of the “city” divisions of information, sorts represented businesses according to their distance from the reader.

Location information can be used to weight the choices of printers that are nearby. Figure 1. shows proximate selection dialogs for printers using three columns: the name of the printer, the location, and a distance from the user. One interface issue is how to navigate dialogs that contain this additional location information. For example, should dialogs use the familiar alphabetical ordering by name or should they be ordered by location. Shown here are (a) alphabetically ordering by name; (b) ordered by proximity; (c) alphabetical with nearby printers emphasized; (d) alphabetical with selections scaled by proximity, something like a perspective view.

Another factor that proximate selection interfaces must take into account is bandwidth requirements. Presenting information that changes, either due to the user moving or the contents of the dialog changing (e.g. other people moving) will cause update network traffic. One approach is to view location information with more or less precision based on the situation. The interfaces in Table 1. are fine-grained – the distance column requires updating for each change in location of the locus. In contrast a coarser-grained view of the same information might show a zone rather than a distance. Driving around town with such a dialog would, for example, change only when the viewer, or the objects in the selection dialog, crossed the city limits.

Proximate selection may also be used to choose virtual objects. Using the PARCTAB voting application, users select previously created ballots either alphabetically or by the current location. This use of proximate selection is helpful when ballots are referenced at particular locations – e.g., voting on what snacks to have at High Tea – or when you are meeting with a group that has just created a ballot.

User interfaces for proximate selection pose some challenges. For example, how can a UI display both alphabetical and proximity information simultaneously. Map imagery may provide a good UI metaphor. Since proximate selection may occur on a mobile host, the UI techniques developed must take into account device capabilities such as screen real-estate and communication bandwidth.

2.2 Automatic Contextual Reconfiguration

Reconfiguration is the process of adding new components, removing existing components. or altering the connections between components. Typical components and connections are servers and their communication channels to clients. However reconfigurable components may also include loadable device drivers, program modules, hardware elements, etc. In the case of context-aware systems, the interesting aspect is how context of use might bring about different system configurations and what these adaptions are.

When a group of people is in one place, the people can easily share the physical objects in that place. For example, people in a meeting room share a table that might hold scattered papers, and a whiteboard with diagrams. To promote similar sharing, we wrote a multi-user drawing program for the PARCTAB which provides a workspace for each room, a sort of virtual whiteboard. Entering a room causes an automatic binding between the mobile host and the room's virtual whiteboard. In this way people in the same room can easily collaborate using the virtual whiteboard. Moving to a different room brings up a different drawing surface. Automatic reconfiguration creates the illusion of accessing this virtual object as if it were physical.

Reconfiguration could be based on other information in addition to location, for example, the people present in a room. If a project group is meeting then the project whiteboard is active. This change makes virtual whiteboards more powerful than their physical analogues since a virtual whiteboard can persist from meeting to meeting, and can follow participants from room to room.

Contextual reconfiguration might also include operating system functions: for example, leaving the disk spinning when the mobile has a power connection. Schilit and Duchamp [8] describe how an operating system can use the memory of nearby idle computers for backing store, rather than swapping to a local or remote disk. The context of use, i.e., the hosts in the vicinity, define the system configuration, and when hosts change location, the configuration adapts accordingly.

Systems that reconfigure based on context are subject to the same problems faced by reconfigurable systems in general. In addition, if the context is changing rapidly it may be distracting to the user or impractical (due to performance) to adapt to every change. Also, certain adaptions may confuse users, particularly if the context is incorrectly reported, if the user is unaware of what context an application considers relevant, or if the context changes during use. Future work should address these issues.

2.3 Contextual Information and Commands

People's actions can often be predicted by their situation. There are certain things we do when in the library, kitchen, or office. Contextual information and commands aim to exploit this fact. Queries on contextual information can produce different results according to the context in which they are issued. Similarly, context can parameterize "contextual commands," for example, the print command might, by default, print to the nearest printer.

The location browser is a PARCTAB application that views a "location-based filesystem." Directories are named after locations and contain files, programs, and links. When moving from room to room, the browser changes the displayed directory to match the viewer's location. For example, when in an office we see the occupant's finger plan and calendar files. In the public area of our lab we see a general description of the research group, and when near the kitchen we see directions for making coffee and finding supplies. Location directories are writable so anyone in our group can add customized information. A researcher might leave an electronic post-it note for colleagues telling them when she plans to return to the office.

Aside from displaying data files parameterized by the viewer's location, the location browser also runs programs. Contextual commands of this kind may take two forms. First, the appearance of the command itself might change depending on context of use. For example, when in the library the button to invoke a card catalogue database might appear prominently whereas it is normally hidden. Second, a command may appear the same but produce parameterized results. For example the location browser presents a migrate button in it's user interface that appears identical from room to room, but that causes the user's windows to migrate to different host displays depending on the location in which it is invoked.

The location browser contrasts with Scoreboard [15], an application that takes advantage of large (3x4 foot), stationary, displays in public areas. The program monitors which users pass by and are in the vicinity and then shows information that they are interested in, e.g., the latest hockey scores. A third example of exploiting contextual information is Lamming and Newman's activity-based information retrieval [6]. In this application, information entered into the computer is tagged with context keys facilitating future retrieval by using those keys.

Category	Example
Date and time	`after April 15`
	`between 10 and 12noon`
Location	`in room 35-2200`
Co-location	`with {User Adams}`
	`with {Type Display} having {Features Color}`

Table 3: Predicates for Context-Triggered Actions

Contextual information and commands pose many challenges. Businesses and government agencies would find it profitable to export contextual information and commands in order to inexpensively engage large numbers of potential customers. However, people interacting with third parties need to ensure security and authenticity of the information. Also, personal customizations must somehow coordinate with those provided by third parties.

2.4 Context-Triggered Actions

Context-triggered actions are simple IF-THEN rules used to specify how context-aware systems should adapt. Information about context-of-use in a condition clause triggers consequent commands; something like living in a rule-based expert system! A number of applications can be organized in this way. The category of context-aware software is similar to contextual information and commands, except that context-triggered action commands are invoked automatically according to previously specified rules. A sample predicate set might include the forms in Figure 2.

We have experimented with two context-triggered action applications, Active Badge[1] based "Watchdog" and tab based "Contextual Reminders." The watchdog program monitors Active Badge activity and executes arbitrary Unix shell commands. On startup the program reads a user's configuration file containing a description of Active Badge events and actions. Entries are of the form:

```
badge location event-type action
```

The `badge` and `location` are strings that match the badge wearer and sighting location. The `event-type` is a badge event type: arriving, departing, settled-in, missing, or attention[2]. When a matching event occurs, Watchdog invokes the action with a set of Unix environment variables as parameters. These include the badge owner, owner's office, sighting location, and the name of the nearest host. For example:

```
Coffee   Kitchen  arriving   "play -v 50 ~/sounds/rooster.au"
schilit  *        attention  "emacs -display $NEARESTHOST:0.0"
```

The first example monitors the "coffee" badge—which is attached to the coffee maker in the kitchen—and plays the rooster sound whenever anyone makes coffee. The second starts an Emacs window at a nearby host whenever the attention signal is received.

In addition to generating the normal functions of reminders at certain dates and times, Contextual Reminders permit a fuller description of the situation for when a reminder should occur. A message pops up according to when, where, who and what is with you. For example, the "next time in the library" or the "next time I see Marvin," or "when I'm back at my desk." The PARCTAB implementation of Contextual Reminders pops up a message on the tab when different situations occur. The user may edit the reminder, dismiss it, or ignore it.

The problems of building context-triggered actions include how to balance the requirement of timely execution with the need for predictable behavior, when systems transition between a number of states it may not be desirable to have all the intermediary actions triggered, but delaying too long will make the system seem sluggish. Two problems to be addressed are the expressiveness of the predicate language, and the accuracy and timeliness of the underlying context information.

[1] The Active Badge is a tag that periodically broadcasts a unique identifier for the purpose of determining the location of the wearer.

[2] The badge incorporates a single finger button, and the attention event is reported when the badge button is clicked twice.

3 Related Work

This research is inspired by the ubiquitous computing vision [16] and our experiences with PARCTAB, the first mobile computing system built to explore and exploit context-aware software [1, 7].

Active Badges developed at Olivetti Research Lab (ORL) [14] focused on the hardware design and implementation of badges and networks of infrared receivers. The main software application, an "aid for a telephone receptionist," shows a table of names alongside a dynamically updating display of locations and telephone extensions. Staff wearing badges can have telephone calls directed to their current location. The original ORL system did not take context into account. Badge wearers expressed a desire to control call forwarding using context information: who they are with, where they are, and the time of day. "Personal control scripts" were added to a later version of the system to address this problem [13].

A mechanism for application contextual-customization using the idea of a user global execution environment, was described in [10]. Similarly "computing personae" [2] address how people moving between systems can have a persistent computing environment. Adaption of user interfaces for wireless mobile systems is described in [3]. An application that uses context to tag information and facilitate retrieval is presented in [6]. In contrast, system level contextual reconfiguration has been examined in [8].

Infrastructure issues in location-aware computing are described in [11, 4, 12] and issues of information dissemination for context-aware software is presented in [9]. "Situation awareness" for mobile communication systems is advocated in [5].

4 Acknowledgments

A number of people worked on the location systems in our lab, including Marvin Theimer, Mike Spreitzer, Karin Petersen, David Nichols and Phil James. Thanks to Rich Gold for Figure 2. John Ellis wrote the PARCTAB voting application. David Nichols and Marvin Theimer wrote the Scoreboard application. Thanks to Natalie Jeremijenko and Douglas Terry for comments and approval. Finally, we appreciate Mark Weiser's leadership in pursuit of the Ubiquitous Computing vision.

References

[1] Norman Adams, Rich Gold, Bill N. Schilit, Michael Tso, and Roy Want. An infrared network for mobile computers. In *Proceedings USENIX Symposium on Mobile & Location-independent Computing*, pages 41–52. USENIX Association, August 1993.

[2] Arindam Banerji, David Cohn, and Dinesh Kulkarni. Mobile computing personae. In *Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 14–20. IEEE, October 1993.

[3] David Goldberg and Michael Tso. How to program networked portable computers. In *Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 30–33. IEEE, October 1993.

[4] Andy Harter and Andy Hopper. A distributed location system for the active office. *IEEE Network*, pages 62–70, January/February 1994.

[5] Randy H. Katz. Adaption and mobility in wireless information systems. *IEEE Personal COmmunications*, 1(1):6–17, 1994.

[6] Michael G. Lamming and William M. Newman. Activity-based information retrieval: Technology in support of personal memory. In F.H. Vogt, editor, *Personal Computers and Intelligent Systems.*, volume A-14 of *IFIP 12th World Congress. Proceedings of Information Processing 92*, pages 68–81. IFIP, Elsevier Science Publishers (North-Holland), 1992.

[7] Bill N. Schilit, Norman Adams, Rich Gold, Michael Tso, and Roy Want. The PARCTAB mobile computing system. In *Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 34–39. IEEE, October 1993.

[8] Bill N. Schilit and Daniel Duchamp. Adaptive remote paging for mobile computers. Technical Report CUCS-004-91, Columbia Univ. Computer Science Dept., February 1991.

[9] Bill N. Schilit and Marvin M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 1994. to appear.

[10] Bill N. Schilit, Marvin M. Theimer, and Brent B. Welch. Customizing mobile application. In *Proceedings USENIX Symposium on Mobile & Location-Independent Computing*, pages 129–138. USENIX Association, August 1993.

[11] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment. In *Proceedings of the Fourteenth ACM Symposium on Operating System Principles*, pages 270–283, Asheville, NC, Dec 1993. SIGOPS, ACM.

[12] Mike Spreitzer and Marvin Theimer. Scalable, secure, mobile computing with location information. *CACM*, 36(7):27, July 1993. In Special Issue, Computer-Augmented Environments.

[13] Row Want and Andy Hopper. Active badges and personal interactive computing objects. *IEEE Transactions on Consumer Electronics*, 38(1):10–20, Feb 1992.

[14] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, Jan 1992.

[15] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.

[16] Mark Weiser. Some computer science issues in ubiquitous computing. *CACM*, 36(7):74–83, July 1993. In Special Issue, Computer-Augmented Environments.