Software Design Issues for Ubiquitous Computing

Gregory D. Abowd
College of Computing &
Graphics, Visualization and Usability Center
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

The defining characteristic of ubiquitous computing is the attempt to break away from the traditional desktop computing paradigm and move computational power into the environment that surrounds the user. Increasing miniaturization brought on by advances in areas such as VLSI design means that we will soon be able to instrument our environments in order to realize the dream of ubiquitous computing. The promise of ubiquitous computing, however, is that the increased pervasiveness of computation will lead to less intrusive and more valuable services to the end user. For this dream to be realized, we need to accompany hardware advances with advances in software technology. In this brief paper, we will discuss some software challenges for ubiquitous computing.

Keywords: ubiquitous computing; context-aware computing; automated capture, integration and access

1 Introduction

The interest in ubiquitous computing has surged over the past few years, thanks to some influential writings and plenty of experimental work. The defining characteristic of ubiquitous computing is the attempt to break away from the traditional desktop interaction paradigm, making computational power available to the end-user when and where desired. Rather than force the user to search out and find the computer's interface, ubiquitous computing suggests that the interface itself can take on the responsibility of locating and serving the user.

The history of computing is filled with examples of radical paradigm shifts in the way humans interact with and perceive technology. From the introduction of video display units to the separate inspirations of networked time-sharing, computer graphics and personal computing, the ideas of a few creative people have drastically modified, and for the most part augmented, our own capabilities. The vision of ubiquitous computing —first expressed by Weiser [20] and grounded in experimental work done at Xerox PARC— holds the promise of yet another interaction paradigm shift.

Increasing miniaturization brought on by advances in areas such as VLSI design means that we will soon be able to instrument our environments in order to realize the dream of ubiquitous computing. The promise of ubiquitous computing, however, is that the increased pervasiveness of computation will lead to less intrusive and more valuable services to the end user. For this dream to be realized, we need to accompany hardware advances with advances in software technology. In this paper, we will describe two features of ubiquitous computing applications that demand a software solution: context-awareness; and automated support for capture, integration, and access to live experiences. We will demonstrate the utility of these two features through example applications that we have developed within the Future Computing Environments (FCE) Group at Georgia Tech.[1]

2 Background on our experience

Our views on software infrastructure for ubiquitous computing have been fueled by several years' experience building applications that fit into this new paradigm for interaction. Here we will describe two such projects that will serve as subjects for discussion in the remainder of the paper.

2.1 Classroom 2000

The Classroom 2000 project is investigating the impact of ubiquitous computing on university education [5, 1, 3]. In a traditional classroom, the instructor either writes on a blank whiteboard or uses an overhead projector to display prepared slides that can then be annotated. The problem with using whiteboards or annotating overhead slides is that the student is forced to copy everything the teacher writes into their

[1] Further information on the FCE Group can be found at http://www.cc.gatech.edu/fce.

own private notebook. Otherwise, once the teacher's writing is erased, the information is forever lost. The advantage of electronic media used in the classroom is that there is the chance to preserve the record of class activity.

We have instrumented a single classroom with a large-scale, pen-based electronic whiteboard that enables an instructor to present and annotate a standard lecture, using a blank surface, a prepared presentation or a series of Web pages as the background. Our software captures much of the lecturer's activity and timestamps it. In addition, the room is equipped with digital recording infrastructure, and we automatically generate Web-accessible notes that coordinate the captured lecture notes with the audio/video recordings and the Web URLs accessed by the lecturer during the class. The result is an environment that attempts to relieve students of the burden of copious, and often incomplete, note-taking so that they can engage more directly in the classroom experience.

We distinguish this electronic media from multimedia courseware. The production of multimedia courseware is a time-consuming task. It may take years of effort to produce a high-quality course module. One of the objectives of the Classroom 2000 project is to reduce the content-generation effort by viewing the traditional classroom lecture itself as a multimedia authoring activity.

2.2 CyberDesk

The CyberDesk project is aimed at providing a flexible framework for automatically integrating the behavior of personal information management applications that can reside on the desktop, the network or on mobile hand-held platforms [22, 8, 7]. Software applications often work on similar information types such as names, addresses, dates, and locations. Collections of applications are often designed to take advantage of the potential for integration via shared information. As an example, an electronic mail reader can be enhanced to automatically recognize Web addresses, allowing a reader to select a URL to automatically launch a Web browser on that location. Even more complex and useful integrating behavior is available in a number of commercial suites of applications. CyberDesk is a demonstration system that provides automatic integration of many personal information services with minimal additional programmer effort and maximum user

We recognize the utility from the user's perspective of integrating the behavior of a number of software applications. With the emergence of Web-based applications and personal digital assistants (PDAs), there are even more opportunities to provide integration of software applications. There are some limitations, however, to the current approaches for providing this integration. These limitations impact both the programmer and the user.

From the programmer's perspective, the integrating behavior between applications is static. That is, the behavior must be identified and supported when the applications are built. This means that a programmer has the impossible task of predicting all of the possible ways users will want a given application to work with all other applications. This usually results in a limited number of software applications that are made available in an integration suite.

From the user's perspective, integrating behavior is limited to the applications that are bound to the particular suite being used. Further integration is either impossible to obtain or must be implemented by the user. In addition, the integrating behavior has a strong dependence on the individual applications in the suite. If a user would like to substitute a comparable application for one in the suite (e.g. use a different contact manager, or word processor), she does so at the risk of losing all integrating behavior.

3 Context-aware computing

3.1 Defining the problem

Future computing environments promise to free the user from the constraints of stationary desktop computing. Increased user mobility, a defining dimension of ubiquitous computing, suggests that applications should adapt themselves based on knowledge of location. This location can be position and orientation of a single person, many people, or even of a certain set of devices. Location is a simple example of *context*, that is, information about people or devices that can be used to modify the way a system provides its services to the user community. Location is an example of physical context. Other categories of context include informational (what data is the user focused on), emotional (how a user feels), intentional (what does the user want to do) and historical (what is the record of context over time). Context-aware computing aims to provide maximal flexibility of a computational service based on real-time sensing of any of these forms of context.

Context-awareness is not unique to ubiquitous computing. For example, explicit user models used to predict the level of user expertise or mechanisms to provide context-sensitive help are good examples used in many desktop systems. With increased user mobility, the physical location of a user changes more often and this information can and should be used to

provide more salient information. Moreover, context-awareness is a critical feature for a ubiquitous computing system because important context changes are more frequent. In a ubiquitous computing environment it is likely that the physical interfaces will not be "owned" by any one user. When a user owns the interface —as is usually the case with personal digital assistant or a laptop computer— over time this interface can be personalized to the user. Context can be useful in these situations, as has been demonstrated by location-aware computing applications. When the user does not own the interface, it is likely that the same physical interface will be used by many people. Any single user (or group of users) will prefer to have the interface personalized for the duration of the interaction. Context-awareness will allow for this rapid personalization of computing services.

3.2 Examples

We have been accustomed to one form of this rapid personalization for many years. Most computer laboratories with networked computers have been set up to allow different users to login and use a number of different (though usually somewhat homogeneous) machines as if each machine were the user's very own. Another example more in line with the intentions of ubiquitous computing is used in Classroom 2000. When a lecturer steps up to the electronic whiteboard at the beginning of class, the board automatically initializes itself for that lecturer to use for the correct class. In the future, when we introduce student note-taking devices, we would want students to be able to pick up any device and have the device rapidly personalized to the specific student in the correct class.

There are many examples of location-aware computing. We have designed a mobile, hand-held tour-guide, Cyberguide, to facilitate indoor and outdoor tours on the Georgia Tech campus [2]. Other seminal work on location-aware systems occurred at Olivetti Research Labs, developers of the Active Badge location system [17], and at PARC, through the PARCTab system [18]. A more general programming framework for describing location-aware objects was the subject of Schilit's thesis [15].

The CyberDesk project described above is an example of informational context. We are now looking at extending the context inferencing engine to support applications in which knowledge of the people and place causes automatic modification of services displayed on mobile displays [7]. Similar work using informational context to integrate desktop applications has been reported by Apple [4] and Intel [13].

Advances in computational perception [9] and affective computing [14] are making it possible for us to consider the possibility of enabling emotional and intentional context either by instrumenting the environment to perceive information about its occupants or by attaching wearable sensors to the users themselves.

3.3 Software challenges

The general mechanism for context-aware computing is summarized in the following steps:

1. Collect information on the user's physical, informational or emotional state.

2. Analyze the information, either by treating it as an independent variable or by combining it with other information collected in the past or present.

3. Perform some action based on the analysis.

4. Repeat from Step 1, with some adaptation based on previous iterations.

The software challenge is to create a context-inferencing engine that is general and scalable. We have attempted to build such an engine in CyberDesk. In the personal information management domain, we want data highlighted in one service to trigger possible actions in any relevant service that can use that data. We have not tested the context-inferencing engine outside of this application domain, so we do not know how general it is. Also, there is still too much extra effort for the programmer to make known to the system what contextual information a given service generates and consumes. Our solution is scalable, but only in the sense that there is no limit to the number of services that can be added. The problem is that informational context will trigger all possible related services, and not necessarily the most salient ones. We still need to find a solution which scales in terms of the number of services supported while still being able to produce a shortlist of timely suggestions for action.

Context is about many things —where someone is located, who is around, what time it is— and it is important to be able to take in as much contextual information to tailor the behavior of a given service. We have explored the issue of combining contextual information in CyberDesk, but our approach has been somewhat primitive. More subtle ways of combining context in order to discern higher-level information about a user (Is she bored? What is the focus of attention of the group in class at this moment?).

4 Automated capture, integration and access

4.1 Defining the problem

One of the potential features of a ubiquitous computing environment is that it could be used to record our everyday experiences and make that record available for later use. Indeed, we spend much time listening to and recording, more or less accurately, the events that surround us, only to have that one important piece of information elude us when we most need it. We can view many of the interactive experiences of our lives as generators of rich multimedia content. A general challenge in ubiquitous computing is to provide automated tools to support the capture, integration and access of this multimedia record. The purpose of this automated support is to have computers do what they do best, record an event, in order to free humans to do what they do best, attend to, synthesize, and understand what is happening around them, all with full confidence that the specific details will be available for later perusal.

4.2 Examples

The Classroom 2000 project is mainly concerned with capture, integration and access in support of lecture-based education. The many streams of activity in a typical lecture —what is spoken, what is seen, what is written down on a whiteboard and what is shown on public displays— are combined to provide a rich interactive experience that is becoming increasingly more difficult to capture using traditional pen and paper notes. Some of the Cyberguide prototypes created summaries of when and where a visitor traveled on campus and preserved images taken with a camera or comments made by the visitor that were attached to this temporally- and spatially-indexed travel diary.

Other research teams have used this same notion of capture, integration, and access to facilitate collaborative or personal experiences. Work at Xerox PARC focused on capturing technical meetings to support summarization by a single scribe who was often not well-versed in the subject of the meetings [11, 12]. More work at PARC (the Marquee system [19], together with work at Hewlett-Packard (the Filochat system [21], Apple [6], and MIT's Media Lab [16] demonstrates the utility of personal note-taking with automatic audio enhancement for later review.

4.3 Software challenges

We will highlight three challenges dealing separately with the capture, integration, and access phases.

Capture

Most interfaces today lack interaction *transparency*, one of Weiser's defining characteristics of a ubiquitous computing system [20]. Interaction transparency applies to the system's interface and reflects the conscious efforts and attention the system requires of the user, either for operating it or for perceiving its output. Transparency during the capture phase is a very important consideration, with both positive and negative ramifications.

On the positive side, we want to push for maximal transparency. For example, in Classroom 2000, teachers are not usually motivated enough to spend extra time before, during and after a lecture interacting with complex equipment to record their lecture for the benefit of students. Capturing the lecture needs to be as simple as picking up a pen and beginning to talk. The burden should be on the system, not the user, to encode the natural activities of a collaborative experience, so that it can be properly indexed to facilitate later review.

We have spent a lot of energy to increase the *syntactic transparency* in Classroom 2000. For example, simply knowing the class schedule in a room removes a lot of initialization tasks that were required of the lecturer. The teacher simply opens up the electronic whiteboard application, types in a title for the lecture, authenticates herself with a username and password and clicks on the "Begin lecture" button. At the end of class, closing the application will automatically set in motion the post-production process that creates the audio/video-enhanced Web notes without any further interaction from the teacher. We have not, however, provided for any level of *semantic transparency* in the system. We would like to detect some higher-level structure in a lecture to facilitate integration and access. For example, a better integration scheme would use more than timing information, linking what is written with the best place in the audio stream that is related to the writing. This is a main software challenge for future automated capture environments.

On the negative side, too much transparency can be problematic, and users must be made aware of what is captured, who will have access to it and why. In addition, there should be ways for the users to control what is being captured and remove segments that they would rather not save. Knowledge of what is being captured can also help the users to adapt their own note-taking behaviors and allow them to be more effective with their time.

Integration

One reason why videotaped lectures are considered

ineffective is that they do not lend themselves to random access and simple browsing. We can increase the value of a simple videotape by providing some way to randomly access significant parts of it. Exploiting the relationship between different captured streams is an integration problem. Once the streams have been captured we are faced with the problem of determining the relationship between different streams and how to integrate them into a presentation of the lecture. All streams have at least one inherent relationship: time. We can easily exploit this relationship in developing a presentation of multiple streams. When one stream is positioned at time t we can position all of the other streams at time t as well. We can use streams to serve as indices into other streams. For example, when viewing a lecturer's notes written on the electronic whiteboard, each individual pen-stroke can serve as an index into the audio recording for the class.

Since time is the common element between streams, we must ask the question, how tightly coupled must the streams be for useful review? We will consider this question with respect to the stream of pen-strokes produced by the teacher using Classroom 2000. Different choices for the granularity of integration provide for different access capabilities. There is a continuum for this granularity, where at one extreme, the coarsest granularity is at the level of each lecture. We provide a single entry point into a stream that is its beginning. This is similar to having a tape recorder without any memory settings; you can only play the tape from beginning to end and there is no way to directly access a particular segment of the recording. At the other extreme, since we have recorded all pen-strokes, we know exactly when every pixel was recorded. Selecting any part of a pen-stroke can take you to the exact time during the lecture when that pixel was drawn. In between these two extremes are some more useful levels of granularity that we have investigated, including slide-level and word level.

An interesting software challenge is to provide mechanisms for determining the correct level of integration between various recorded streams and to provide the flexibility for the user to dynamically adjust the level of integration during the access phase.

Another challenge we have only begun to address is to use information other than time as an indexing agent. Some examples are content-based multimedia retrieval techniques [10] and the use of physical artifacts to index into recorded memories.

Access

Our initial goals with Classroom 2000 were modest. We wanted to display teacher notes with audio augmentation at some useful level of integration. With successful deployment of such a system, we noticed the need to be able to support more streams of information generated during lecture, such as URLs visited by a Web browser, arbitrary video clips shown during class, execution of arbitrary computer programs, and a variety of student-generated notes. The integration techniques discussed above are effective for visualizing one stream of information, such as the notes, and using that visualization as an interface to another stream, such as the audio or video. While it was clear how we could introduce more captured streams into the class, it was not at all clear how to present all of those streams of information effectively to the student in a way that supported their goal of review. We need more than inter-stream integration.

To help us understand some of the visualization issues, we began an effort to prototype interfaces that supported the numerous streams we envisioned capturing. The immediate problem was the tendency to overload the screen with far too many streams, making it difficult for the user to understand the relationships between them and even harder to know how to proceed to the point of the lecture that is of interest for review.

Any visualization technique for multiple streams must meet two challenges. It must scale to support many streams and it must provide the student with the ability to scan through an entire stream quickly in order to locate some point of interest. We have investigated a number of techniques to support multiple stream visualization that are scalable and support the "browsing to pinpoint" review behavior [3]. One of our more successful interfaces uses a central timeline controller decorated with pointers to artifacts from various streams (a new slide from the electronic whiteboard or a newly visited URL).

5 Conclusions

From a hardware perspective, the age of ubiquitous computing is at hand. From a software perspective, we have a long way to go toward realizing even part of Weiser's initial vision of ubiquitous services with transparent interfaces. In this paper, we have tried to identify two common features of ubiquitous computing applications —context-awareness and automated capture, integration, and access. By identifying these features, we hope to motivate software designers to build general support for others to rapidly prototype various ubiquitous computing applications. By doing so, we will be able to bootstrap the necessary empirical work to determine how this emerging interaction

paradigm will impact our everyday lives.

Acknowledgments

The work reported in this paper has been sponsored by the National Science Foundation (faculty CAREER grant IRI-9703384), the DARPA EDCS program, Motorola Corporation, Sun Microsystems, Hewlett-Packard and Proxima Corporation. Dr. Abowd would like to thank the numerous colleagues (students and faculty) who have contributed to the work of the Future Computing Environments Group since 1995, especially Anind Dey, Jason Brotherton, and Daniel Salber, whose work is directly reflected in the content of this paper.

References

[1] G. D. Abowd, C. G. Atkeson, J. Brotherton, T. Enqvist, P. Gulley, and J. LeMon. Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. In *Proceedings of the 1998 conference on Human Factors in Computing Systems — CHI'98*, 1998. To appear.

[2] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, 3:421–433, 1997.

[3] G. D. Abowd, J. Brotherton, and J. Bhalodia. Automated capture, integration, and visualization of multiple media streams. In *Proceedings of the 1998 conference on IEEE Multimedia and Computing Systems*, 1998. To appear.

[4] Apple Research Laboratories. Apple data detectors homepage. Available at http://www.research.apple-.com/research/tech/AppleDataDetectors/, 1997.

[5] J. A. Brotherton and G. D. Abowd. Rooms take note: Room takes notes! In *Proceedings of the 1998 Spring AAAI Symposium on Intelligent Environments*, 1998.

[6] L. Degen, R. Mander, and G. Salomon. Working with audio: Integrating personal tape recorders and desktop computers. In *Proceedings of ACM CHI'92 Conference*, pages 413–418, May 1992.

[7] A. Dey. Context-aware computing: The cyberdesk project. In *Proceedings of the 1998 Spring AAAI Symposium on Intelligent Environments*, 1998.

[8] A. Dey, G. D. Abowd, and A. Wood. Cyberdesk: A framework for providing self-integrating context-aware services. In *Proceedings of the 1998 Intelligent User Interfaces Conference — IUI'98*, pages 47–54, 1998.

[9] I. Essa and A. Pentland. Facial expression recognition using a dynamic model and motion energy. In *Proceedings of the International Conference on Computer Vision*, pages 360–367. IEEE Computer Society, Cambridge, MA, 1995.

[10] M. T. Maybury, editor. *Intelligent Multimedia Information Retrieval*. AAAI Society and MIT Press, 1997.

[11] S. Minneman, S. Harrison, B. Janseen, G. Kurtenbach, T. Moran, I. Smith, and B. van Melle. A confederation of tools for capturing and accessing collaborative activity. In *Proceedings of the ACM Conference on Multimedia — Multimedia'95*, November 1995.

[12] T. P. Moran, L. Palen, S. Harrison, P. Chiu, D. Kimber, S. Minneman, W. van Melle, and P. Zelweger. "I'll Get That Of the Audio": A case study of salvaging multimedia meeting records. In *Proceedings of ACM CHI'97 Conference*, pages 202–209, March 1997.

[13] M. Pandit and S. Kalbag. The selection recognition agent: Instant access to relevant information and operations. In *Proceedings of Intelligent User Interfaces '97*. ACM Press, 1997.

[14] R. Picard. *Affective Computing*. MIT Press, 1997.

[15] W. N. Schilit. *System architecture for context-aware mobile computing*. PhD thesis, Columbia University, 1995.

[16] L. J. Stifelman. Augmenting real-world objects: A paper-based audio notebook. In *Proceedings of ACM CHI'96 Conference*, pages 199–200, April 1996. Short paper.

[17] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[18] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, J. Ellis, D. Goldberg, and M. Weiser. The PARCTAB ubiquitous computing experiment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, March 1995.

[19] K. Weber and A. Poon. Marquee: A tool for real-time video logging. In *Proceedings of ACM CHI'94 Conference*, pages 58–64, April 1994.

[20] M. Weiser. The computer of the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[21] S. Whittaker, P. Hyland, and M. Wiley. Filochat: Handwritten notes provide access to recorded conversations. In *Proceedings of ACM CHI'94 Conference*, pages 271–277, April 1994.

[22] A. Wood, A. Dey, and G. D. Abowd. Cyberdesk: Automated integration of desktop and network services. In *Proceedings of the 1997 conference on Human Factors in Computing Systems — CHI'97*, pages 552–553, 1997. Technical note.