INCA: Architectural Support for Building Automated Capture & Access Applications

Khai N. Truong and Gregory D. Abowd
College of Computing & GVU Center
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280 USA
+1 404 545 1036
{khai, abowd}@cc.gatech.edu

ABSTRACT
Applications that automatically capture some details of a live experience and provide future access to that experience are increasingly common in the ubiquitous computing community. However, there remains a largely unexplored design space of potential new applications, and very few of the previous systems have been able to evolve in functionality over an extended period of time. To overcome these challenges, we present a distillation of the essential architectural features of an automated capture and access application. We introduce a toolkit, Infrastructure for Capture and Access (INCA) that encourages a simplified model for designing, implementing and evolving capture and access applications. We validate the utility of INCA through three sample applications that show variety within the wider design space and accessibility of the toolkit for others.

Keywords
Ubiquitous computing, capture and access applications, architecture, toolkit, application development.

INTRODUCTION
In his 1945 Atlantic Monthly article, Vannevar Bush described his vision of the *memex*, noting that a "record … must be continuously extended, it must be stored, and above all it must be consulted" [3]. Our daily lives provide us with a great deal of records and memories that we want to access at some point in the future. Inspired by the vision of Bush and the progress of ubiquitous computing over the past decade, many researchers have demonstrated devices and applications to support the automated capture of live experiences and the future access of those records (see [31] for a full review). The purpose of every capture and access application is to preserve some portion a live experience so that it can be consulted in the future. Broadly speaking, a capture and access application needs to:

- preserve relevant details from an experience;
- mark associations between a variety of related captured artifacts; and
- provide effective interfaces to review past experiences in a way that suits some human concern.

Though there are many demonstrations of automated capture and access applications in varying domains, such as classrooms [1, 7, 21], meetings [5, 22, 26, 28, 33, 34], and conferences [11], there remains a largely unexplored design space of potential new applications. Furthermore, there has been relatively little evaluation of these systems under authentic use, with the notable exception of Tivoli [20] and eClass [2], largely because these demonstrations are hard to build and even harder to maintain and evolve over the course of a longitudinal study of use.

To facilitate research in ubiquitous computing, advances are necessary to improve the tools we provide ourselves and other creative designers who wish to realize and improve upon the visions of Bush and Weiser. Previously, we have worked on tools to support context-aware computing [9, 10] and human-assisted error correction resulting from recognition-based interfaces [17, 18]. In these previous cases, our method has been to present the relevant design abstractions for a well-defined class of applications, develop an architectural solution to support the design and construction of these applications, implement a toolkit that embodies these abstractions and then validate the abstractions, architecture and toolkit by developing interesting and complex applications within the design.

We apply this research method for the class of automated capture and access applications. Specifically, we present an overview of the essential features of capture and access applications that define the relevant design space. We relate the kinds of problems developers have with the construction task for capture and access applications. We present those characteristics common across all such

*Submitted for review to **UIST 2002**.*

applications, resulting in a language for design. There is an explicit architecture for these applications that is presented and a toolkit, INCA (Infrastructure for Capture and Access) that embodies the relevant abstractions and architecture. INCA is successful to the extent that it:

- lowers the barrier for others to build capture and access applications;
- inspires the construction of a larger variety of applications previously unexplored; and
- facilitates evolution of applications as they are fine-tuned to meet the users' needs.

Overview

We begin by outlining the essential features of capture and access applications, a distillation of our extensive experience building successful and not-so-successful systems. The rest of the paper focuses on the definition and use of the INCA toolkit. We begin with a discussion of the assumed architecture of a capture and access application that supports and a number of important programming abstractions and run-time features that simplify many of the challenges in constructing applications in this domain. We end the paper with a presentation of three sample applications meant to validate our claim that INCA achieves the goals of lowering the barrier to entry, inspiring creativity in the application space and facilitating evolution. We conclude with a discussion of related work and future challenges.

ESSENTIAL FEATURES OF CAPTURE & ACCESS APPLICATIONS

Our own investigation of automated capture and access of live experiences began with the eClass project (formerly known as Classroom 2000) [1]. In retrospect, the relative success of this project in the classroom and as a research platform led us to investigate the capture and access of informal and serendipitous meetings, software architectural discussions, distributed meetings, and domestic settings. For a detailed review of our experience, consult [31]. Through this experience, we have defined five domain-specific aspects of the capture and access application space that help designers tailor specific solutions:

1. ***Who*** are the users during capture and access? It is important to identify how many users are involved in the separate capture and access phases, what roles they have in this application domain, and whether captured records are public, private or a mixture.
2. ***What*** is captured and accessed? An experience is defined in terms of some set of artifacts that are manipulated and information streams that are generated. Which of these artifacts and streams are important to review later on, and what level of fidelity is required between the live experience and the playback of the captured experience?
3. ***When*** does capture and access occur? How often does the captured experience occur and is there any pattern that predicts when it occurs? How much time lag is there between the live experience and the expected time of access?
4. ***Where*** does capture and access occur? Does capture or access occur in a well-defined location or set of locations? Is mobility during capture or access important?
5. ***How*** is capture and access performed? What devices and tools are required during the live experience and must be instrumented in order to facilitate capture of activity? What devices are used to access a past experience?

These aspects help define a design space for capture and access applications that shows how previous research relates as well as highlights interesting gaps in the design space that remain to be explored. Some examples of these gaps include capture of distributed experiences, capture with user mobility, context-based access of captured experiences and long-term access of captured data.

In addition to understanding the important features of the design space, we have also gained an appreciation of the importance of structuring a capture and access application. A primary reason why our classroom research succeeded was because we could evolve the system over a 3.5-year period of evaluation, aided considerably by the architectural decisions made early on. The system was structured into four phases: pre-production, live capture, post-production/integration, and access [1]. This separation of concerns allowed for parts of the system to be altered without globally affecting the entire system. As our exploration of capture and access continued beyond the classroom, we learned several other important architectural lessons:

- Information must be stored (or persist) until it is later accessed. This may seem obvious, and is implied in the post-production/integration phases of eClass, but it is often overlooked as a design issue.
- The phases of eClass imply a sequential ordering of activities, but it is better to consider the functional components of the overall architectural solution. Therefore, we have capture, storage and access as important building blocks for all capture and access applications, with no implied ordering of when they occur relative to each other.
- The post-production/integration activity is further separated into concerns pertaining to: storage, as discussed above; transduction (or transformation) into different data types; and integration, in which relationships between separately captured streams cause the multiple streams to be delivered collectively during access.

- Information integration occurs due to different contextual relationships between captured streams. Examples are streams captured at the same place and time, or covering the same topic, or captured by the same person.
- Access happens on varying time-scales, depending on when information is accessed relative to when it was captured. Therefore, different forms of access interfaces and services are desired, such as a summarized view that allows the user to drill down over summaries of long-term captured data.

THE INCA TOOLKIT

In order to allow designers to focus on the essential features of a capture and access application, we need to provide abstractions and tools that fit the architectural insights presented above. INCA provides a small set of key architectural abstractions, which we explain below. There are additional features of INCA that simplify other aspects of application development that stem from the inherent distributed nature of these applications, and common data types and features that allow programmers and end-users to inspect and control the run-time system.

The Architecture of an INCA application

As motivated above, INCA supports the following separate functional components of a capture and access application:

- Part of the system is responsible for the **capture** of information as streams of data.
- Part of the system is responsible for the **storage** of attribute-tagged (representing contextual metadata) information.
- When information needs to be converted into different formats and types, part of the system must **transduce** the information.
- Part of the system provides **access** to multiple related, or integrated, streams of information that are gathered as response to context-based queries.

From the design perspective, any capture and access application is expressible in terms of these basic functions. For any given application, there may be more than one instance of each of the above functions (e.g., the personal audio loop application discussed below provides a capture component for each room requiring audio capture).

From the implementation perspective, the INCA toolkit provides a direct way to translate the design into executable form. For each functional component above, INCA defines an interface and an encapsulated module that a programmer implements and instantiates as part of the application code. As we will demonstrate later, this results in more rapid application development because the similar structure of all applications eliminates many of the difficulties of coordinating distributed components.

Specifically, the part of the system responsible for the capture of information implements the *Capturer* interface and instantiates a *CaptureModule* object. Similarly, there are *Storer*, *Transducer*, and *Accessor* interfaces and *StorageModule*, *TransductionModule*, and *AccessModule* object classes for those respective portions of the system. We wrap support for the integration of information directly into support for the access of the information, such that when information is requested, related streams of information are jointly provided.

Rather than needing to worry about issues pertaining to data distribution (e.g., what IP address devices are running on, what network protocols are supported, etc.) INCA provides developers with a network abstraction. This abstraction allows application developers to create applications that function without needing to know how the different parts communicate with one another. These specialized network modules only need to be made aware of a *Registry* running at a well-known location. The *Registry* object maintains a list of the available modules that handle the capture, storage, transduction and access of information. This object completely abstracts the communication between all relevant applications and devices.

In Figure 1, a simple capture and access application is built as a distributed system constructed as separate *Capturer*, *Storer*, and *Accessor* components. It is possible to have any number of components running at the same time (though we only show 3). Additionally, a component does not have to be strictly only a *Capturer* or an *Accessor*, etc. Instead, a self-contained device may be a *Capturer*, *Accessor* and *Storer* all wrapped in a single component.

One example of how INCA simplifies the programming task is seen in the relationship between an *Accessor* and

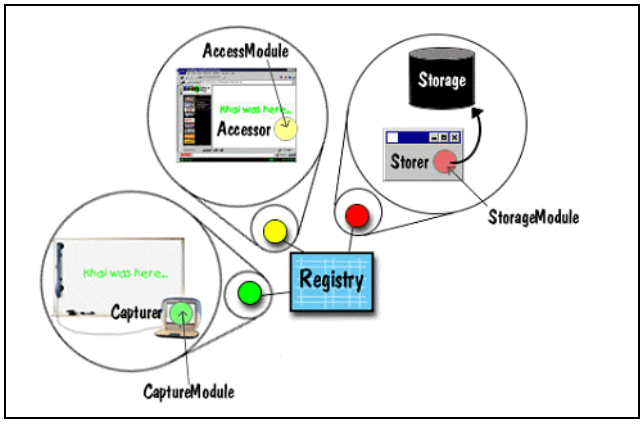


Figure 1. General architecture for systems built using INCA. A *Registry* is running at some well-known location and any number of applications acting as *Capturers*, *Accessors*, *Storers*, or *Transducers* can connect to it and share captured information through instances of specialized networked modules (such as a *CaptureModule*, *AccessModule*, etc.).

the rest of the run-time system. An *Accessor* makes context-based queries (e.g., deliver all data owned by "Khai" originating from "CRB 381" in the past week) but the application programmer does not need to know where any of this captured data resides. The run-time system of INCA resolves the query and delivers the information to the requesting *Accessor*.

Additional Features & Reusable Services

In any applications development effort, there are many stakeholders involved. Features were included in INCA to allow designers to develop adaptable systems, for evaluators to better understand uses of the systems, and to better protect privacy for users. We next discuss the implementation details of important features and reusable services supported by INCA.

Support for common data streams

There is much variety in the kinds of data (or information) involved in different applications. Abstracting away the details of the information being captured allows the infrastructure to provide generic support to a variety of applications designed for different domains. By viewing captured data as only raw bytes with tagged attributes, the infrastructure is able to handle all data in the same generic fashion.

In our review of the literature, we have found a set of data types that are typically captured for later review —ink, audio, video and Web visits. There is a tendency for developers to re-implement devices and applications to work with these data types. However, reusable components can be constructed to remove future replicated efforts. INCA includes a library of reusable components to handle the capture, access, storage, and transduction of these specific data types.

Reusable run-time services

Different capture and access applications will often require the same kinds of services. Once a component is implemented in INCA, it can be made available during run-time for a variety of applications. This encourages the development of general-purpose run-time services. We list a few that have already been created in the current INCA distribution.

Attribute-based storage. A *Repository* service is a sample service extending the basic *StorageModule* and *Storer* interface. It provides a relational database and supports the storage and retrieval of any kind of data tagged with attributes. A *Repository* can be launched and left running, so that application developers can have storage performed as an existing service without additional development effort or modification. The *Repository* class can also be extended to meet a specific application need, such as storing only personal information or optimized for a specific captured data type.

Audio capture. Classrooms in major universities are often used for many different reasons, such as: for classroom lectures, project meetings, or private discussions. A common information stream that would be useful to capture, regardless of the application, is audio. In the physical environment, an audio capture service can provide recording services available to all the different capture and access applications in that environment. Later in this paper, we will present the use of an audio capture and access service to provide near-term audio reminders to users.

Attribute-based access & information integration. As mentioned previously, the support for the integration of information is wrapped into the support for the access of the information. Information is integrated based on how it is requested through context-based queries. In its simplest form, the query match is based on attribute name-value pairs and can grow to include more general data retrieval operations that more effectively filter and mine large distributed repositories. Various temporal and spatial integration techniques have been explored in the past [1, 4]. These techniques, and more, can now be created as reusable integration services.

Scalability

Issues pertaining to data distribution are completely abstracted away by a *Registry* object and its relationship to the other architectural modules. While a few systems operate on a single stand-alone device, most systems are built using multiple networked devices. As a result, while INCA relieves the programmer from having to worry about most of the concerns of distribution, there is a potential for scalability problems when the number of devices in a system is not known ahead of time and can continue to grow as devices are introduced into the environment.

We perform a simple network filtering method to minimize the amount of bandwidth consumption. First, different parts of the system are responsible for different roles, such as capture, storage, etc. When applications instantiate specialized modules (such as a *CaptureModule*), the modules are viewed as either an information produce, consumer or both. This categorization is used to cut back on some needless network traffic. Additionally, components subscribe for specific information they want to consume or publish a set of attributes describing the information they are capable of producing. Thus, when information is requested, only information producers are checked to see if the kind of information requested is among what they can provide. Information is delivered across the network only after these two checks are satisfied. Similar filtering is done to reduce network bandwidth for capture and transduction activities.

Observing the run-time state

An *ObserveModule* is available to allow components implementing an *Observer* interface to gain access to the state of the system. An *Inspector* component is available in the library of reusable components to allow users to quickly view the entire state of the system. These

services that reflect the run-time state are useful for a variety of stakeholders.

First, there is an inherent concern for privacy as capture often involves the preservation of potentially important and personal information. One way to alleviate concern is to provide a way for users to understand the status of the system. Knowing what information can be captured in an environment, and what is currently capturing or not can allow users to adapt their behaviors in that environment.

Second, these observation services allow application developers to build context-aware capture and access applications. In knowing the set of capture and access components available in a particular environment, it is possible to enable applications to leverage the available capture and access services.

Third, system evaluators can use observation services to keep a better log of how the system is used. The actual use of a system in an authentic setting is often different from that in laboratory settings. As ubiquitous computing systems are often built for anywhere-anytime use, it has been a problem in the past to gain an actual understanding of when the system is in use and how. This feature allows the evaluator to build an application to subscribe for changes in the status of the system in terms of when components are introduced, what kind of information it works with and how it manipulates that information.

Controlling capture & access

In addition to being able to observe the run-time state, we also want to control the capture and access of activities. This feature could potentially allow users to turn on or off the capture during a live experience to keep aspects of the experience private. Likewise, in a context-aware system, if a person has a preference of never having any information captured and is the only one in an environment, the system can have all capture temporarily disabled.

Applications implementing a *Controller* interface can use an instantiated *ControlModule* to affect all other components on the system. A *Manager* component is available in the library of reusable components to allow users to quickly view and control the entire state of the system.

APPLICATIONS BUILT WITH INCA

Now that we have defined the key architectural abstractions and other useful services provided by INCA, we must validate all three claims for a toolkit to support research in capture and access:

1. INCA lowers the barrier for development;
2. INCA inspires exploration in the design space; and
3. INCA facilitates evolution of applications.

We present three applications of automated capture and access built using INCA. The first two were designed and developed by the authors and demonstrate exploration of the design space and evolution. The third application was designed and developed independently by a researcher new to the area of automated capture and access. For each application, we will provide a brief motivation, describe how the system was designed and built, and highlight the role of INCA in that exercise. Additional implementations of applications using INCA are discussed elsewhere [29, 30].

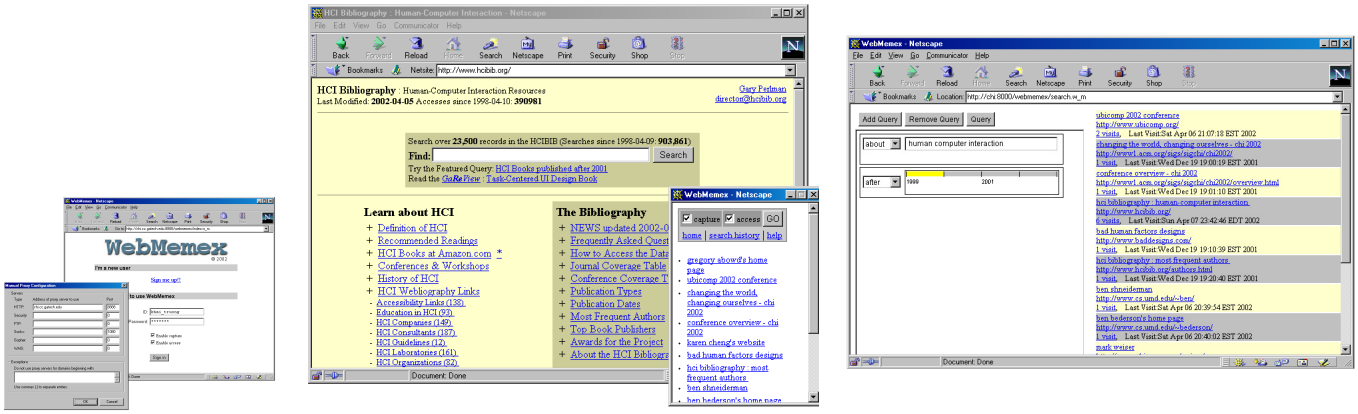
WebMemex

Motivation

WebMemex is an application built to support the continuous capture of a user's Web history. Initially inspired by the value of Web capture in eClass lectures, we wanted to build a more general capture service in support of several access services: the suggestion of related URLs that a user has seen in the past (in the spirit of the Remembrance Agent [25]); an explicit search over a user's complete Web history; and a collaborative filter to suggest relevant Web sites to friends (see [30]). History mechanisms in standard Web browsers are impoverished, requiring users to recall hard-to-remember features of the Web page, such as its URL or title. As we experience the Web, we visit many sites and view many documents; however, when we want to retrieve previously seen information, too often we have either forgotten to bookmark the relevant URLs or the local history mechanism of the browser machine we used some time ago is not accessible.

How WebMemex works

The WebMemex service is provided through an augmented Web proxy server, resolving the problem that existing history mechanisms are local to a single machine. Standard Web browsers can be quickly configured to talk to an HTTP proxy. When a user surfs the Web, her browser will request Web pages from the proxy server. The Web proxy server retrieves the request and serves it to the user on the requesting client browser. If the user has capture and access services enabled, the Web proxy server will react appropriately when the document being returned is of the content type `text/html`.



(a) (b) (c)

Figure 2. The WebMemex Prototype. To use WebMemex, a user simply configures of a browser to talk to a proxy server that captures Web histories and when a user begins a Web surfing session, the browser automatically goes to a screen for the user to sign (2a). Once authenticated, the Web browser captures the user's surfing history and recommends to her URLs she has previously seen that are related (2b). The user may also perform searches over her entire personal surfing history by context (such as when or where she saw the page) and content (such as what the page was about).

If the user wants her session captured, Web visits are tagged with the URL, the title, up to 10 keywords for that Web page, the time that Web page was visited, the IP address of the browser machine, and the user's ID. The captured information is stored in a property-based *Repository* until it is later accessed. Two separate access services are supported implementing different integration techniques. To support the recommendation of related URLs to what the user is currently viewing, Web pages matching on keywords are queried for, and displayed in a small popup window (see figure 2b). To support the search over a user's Web history, information is retrieved based on queries explicitly set by the user. Once found, users can also review the trails surrounding a particular Web page visit; in this scenario, these trails come from a stream of Web visits temporally integrated.

How INCA helps

The Web proxy is constructed as an INCA application. It consists of a single *CaptureModule* that adds a Web visit to the list of previously seen pages. All visits are time-marked, so multiple visits to a Web page are recorded.

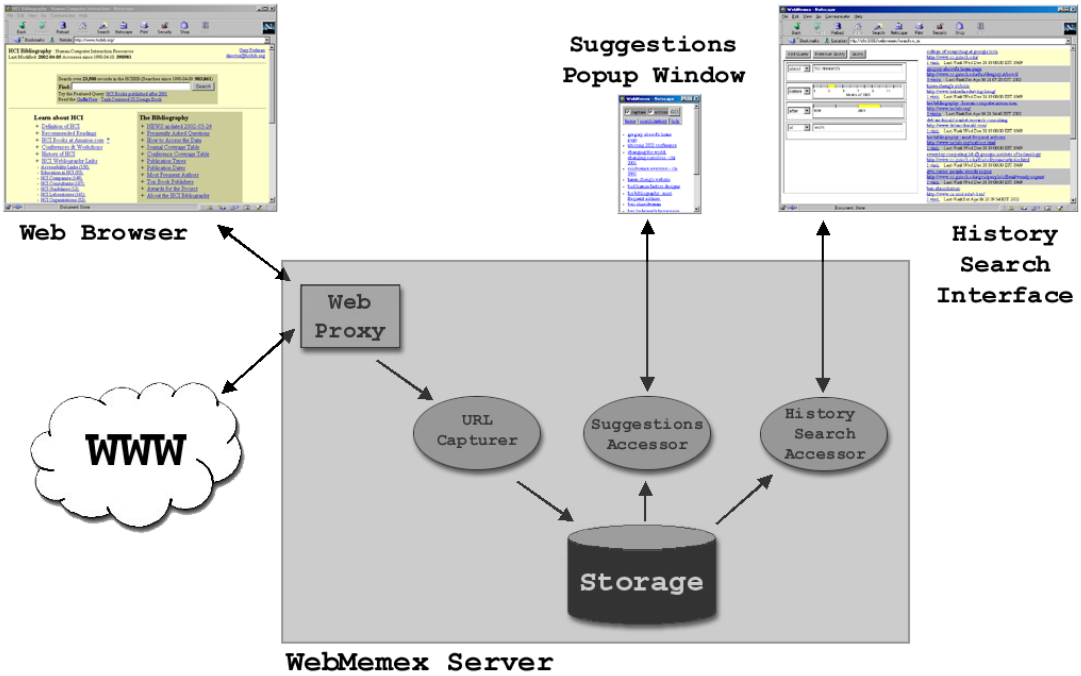


Figure 3. WebMemex architecture. WebMemex is an enhanced Web proxy server to capture Web history and provide access services.

There is a single *StorageModule* (is it an instance of the already defined *Repository* services). The variety of access services (recommendation, search, collaborative filtering) are each implemented as a separate *AccessModule*. The only application logic required for the Web proxy is to be aware of the user's desire for capture and access so the services are invoked only when needed. This system demonstrates information that is captured once but can be useful in many different access scenarios. The different access features explore the integration of information over time and by keyword. In INCA, the different ways to integrate information differ merely in an information query statement; this allows expert designers to explore more sophisticated integration techniques with no more difficulty –similar to the ideas of expert use in interface design.

Personal Audio Loop (PAL)

Motivation

The Personal Audio Loop application explores the near-term capture and access of audio streams to provide users with a quick retrospective memory of recent events. This system can assist with interruption recovery and activity resumption. Unlike a tape recorder, this service continues to capture audio even when playback of previously recorded information is accessed. Near-term audio reminder services such as this have been explored in the context of telephone conversations [8, 16].

How PAL works

We designed an application aimed to provide personal, near-term reminders through the use of a short audio-loop. Initially our prototype was built as an application to run on a single device (laptop) that users would carry with them into meeting rooms (see Figure 4). In this version, audio is continuously recorded and segmented into 4-second chunks. As capture occurs, a semi-persistent storage is made aware of the audio data just captured and is responsible for providing access to components when requested, and deleting the audio segments after they become older than 15 minutes. When users interact with the interface (see figure 4), they are essentially specifying a time-point at which to begin review. Audio is then stitched together for playback.

The simple user interface was inspired by playback capabilities of a modern digital video recorder, such as TIVO and ReplayTV. When a near-term reminder is desired, the user can jump back 30 seconds into the recently recorded audio stream. She can also nudge forward 7 seconds in the event of an overshoot. More sophisticated skimming techniques are currently under investigation and can be easily introduced into the application.

Simplified evolution of PAL with INCA

Though we found this standalone service useful in our own experience, we wanted to eliminate the need for users to carry any device. We wanted to evolve the application and offload more functionality to the

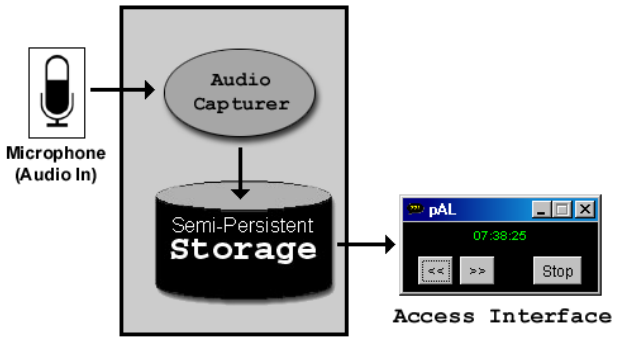


Figure 4. Personal Audio Loop architecture. A near-term audio capture and access reminder system, in which audio from a microphone input is captured and put into a semi-persistent storage (where information lasts for at most 15 minutes). Audio stored in the storage can be accessed by an interface which allows users to jump back 30 seconds at a time in the audio, 7 seconds forward, or to stop (and return to live).

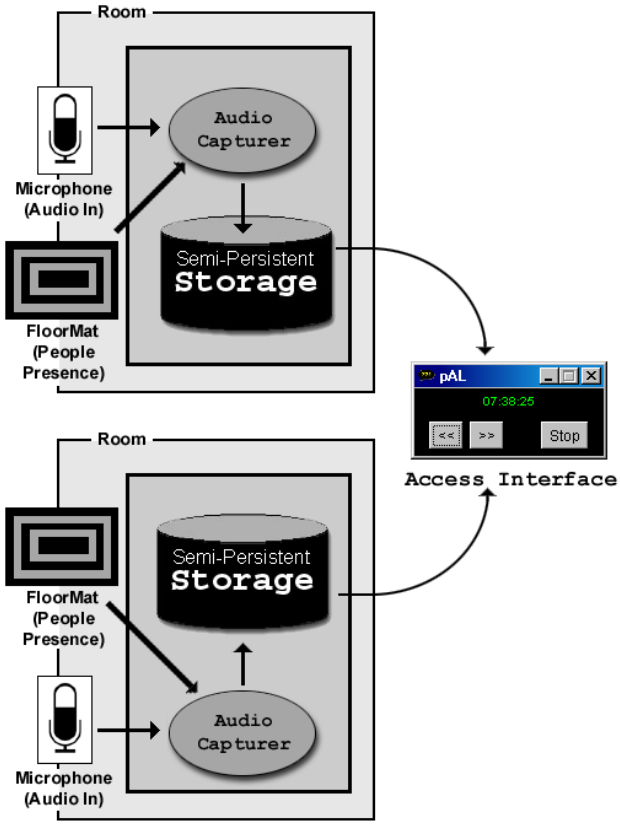


Figure 5. Personal Audio Loop architecture for multiple rooms. Each room is instrumented with microphones and floormats (to identify who are present in the rooms). The Audio Capturer captures audio a semi-persistent storage for that room. Audio is then retrieved by the access interface from different storages (depending on where the user has been in the last fifteen minutes), stitched back and played as it is retrieved.

environment. Recording was distributed to a number of different rooms in our lab environment. Room-level indoor positioning information is then used as context to allow us to tag recorded audio with information of which people were present during any interval of the recording. The effect for a single user is that they can now access any portion of their recent past, regardless of where they were and without having to carry around a recording device.

Even though the original PAL was built as a stand-alone application, it was trivial to convert to this distributed version. The original capture component was modified to tag audio segments with user identities and then deployed as a running service in several locations. The access component was modified to allow for a query containing a user identity. No modifications were necessary for the storage component. A single *Registry* is used to connect all the capture, storage, and access services.

Synchronous Discussion Tool

Motivation

The previous two examples demonstrated relatively novel capture and access applications whose design, construction, and evolution by the authors was facilitated by INCA. One other goal for this work is to lower the barrier for others to build a variety of capture and access applications. To begin validating this claim, INCA was made available to a research group at the University of São Paulo in Brazil. The Intermedia group there is primarily interested in multimedia and hypermedia research, but is interested in looking at capture and access as a means of simplifying the authoring of multimedia documents [23].

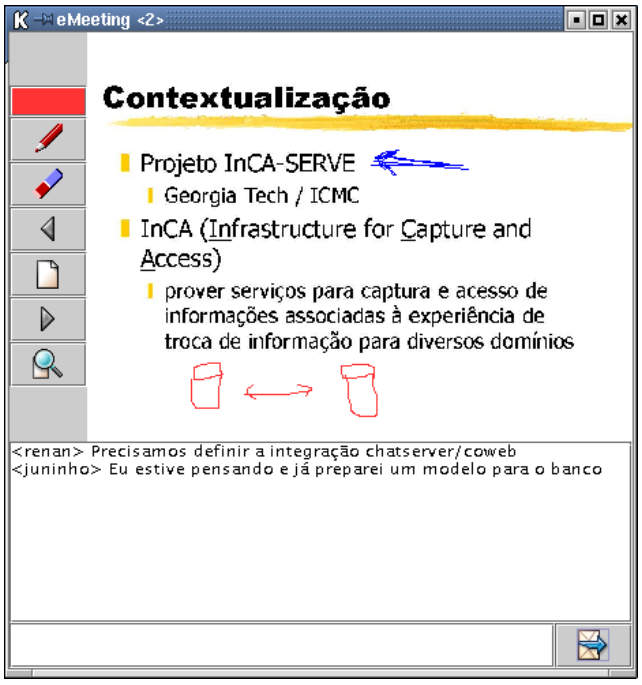


Figure 6. Chat application allowing people to share text and ink messages.

In their research, they have adopted the use of CoWebs [15] as asynchronous discussion spaces, and are investigating how asynchronous discussion spaces influence interesting synchronous discussions between people currently viewing the same page. This has led to the development of a whiteboard and text chat tool in which the discussion is stored back into the discussion space for future reference.

How the chat tool works

Synchronous communication provided in chat tools can be viewed as a capture and access problem, as well. Information created by a user is captured and quickly made available to any application interested in it. The chat tool is connected to the CoWeb discussion space. Any number of users can participate in the discussion; multiple discussions can occur at the same time. Furthermore, each CoWeb page is uniquely identified through a URL. Therefore, a particular chat session can be tagged with the context of chat participants, time and the originating pate from the CoWeb.

How INCA lowers the barrier

This application illustrates the support to help application developers overcome barriers in building simple capture and access applications. There are a number of different ways this application could have been built. This application developer took an approach in structuring the solution different from what we would have done. Our approach would have created two separate capture components for the whiteboard and text chat instead of a single combined component. However, it is encouraging to see that the abstractions presented by INCA are flexible enough to support a variety of solutions in these simple cases. We are confident that INCA will be suitable for general distribution to encourage exploration of the large design space of capture and access.

RELATED WORK

Of the existing related work, most are demonstration vehicles to illustrate the benefit of automated capture and access for a variety of situations, such as the classroom, meetings, and other generalized experiences. The novel aspect of each application pertains to some new interaction much which the designers are most interested in exploring, such as:

- How to capture information using vision [13] [21]
- How to integrate various captured streams [1, 4]
- How to store information in a more fluid manner [12, 14, 24]
- How to access captured information in a more flexible and natural method [32].

A lot of effort and overhead must be taken in order to get the system built before a narrow aspect of the capture and access problem gets explored.

There has been previous work that looks at some of the challenges involved in providing infrastructural support for more general exploration of the potential uses of capture and access applications. The STREAMS work introduced a technique for capturing multiple streams of information as separate, single medium streams that can be temporally integrated [6]. The Tivoli work at Xerox PARC explored how to coordinate multiple applications to work collectively to capture multiple streams of information [19]. Lifestreams [14]and TimeScape [24] are examples of systems which uses time to coordinate the display of all the documents people create. Presto, on the other hand, describes a placeless method for storing information that allows related pieces of information to be grouped together and retrieved [12]. Multimedia documents demonstrate how SMIL (and the use of time) can coordinate the playback of the captured experience. This previous body of work presents key features that are desired from a single infrastructure for capture and access applications [27].

The work we have presented in this paper provides all the functionalities mentioned here —integrating these features into a single framework that manages all the implementation details involved. Beyond this goal, our work allows designers to decompose the development process into smaller parts that are easier to attack separately, and enables the development of reusable and customizable building blocks for further investigation. In doing so, we present a community of application developers with the opportunity to explore the construction of applications in a variety of domains —to build applications that explore the full range of possibilities in the capture and access design space.

CONCLUSIONS

Many remain inspired by the visions of Vannevar Bush and Mark Weiser. We have entered an important stage in ubiquitous computing research. While there is still plenty of room for innovative technological advances, we are to the point where it is important to begin to understand how ubiquitous computing impacts the human experience in practice. However, serious improvements to the tools we use are required in order for major advances in the human-centered research agenda of ubiquitous computing. We cannot observe what we cannot build and use reliably.

In this paper, we have targeted the specific challenge of building applications that support the automated capture of live experiences for later access. There is plenty of motivation to believe that this class of application can provide meaningful and useful services to end users, but there is not enough evidence to validate the claim. This is because capture and access applications are hard to build and keep operational, resulting in a limited variety of demonstrational applications and even more limited empirical observation of use.

We presented key architectural insights into the creation of capture and access applications. Designing in terms of these architectural features —capture of attribute-tagged data streams, storage, transduction and access of related and integrated streams— allows a designer to focus on key distinguishing features of any capture and access application. We introduced the INCA toolkit as a means of transforming high-level designs into implementations that hide details of distribution and data from the programmer. To validate the effectiveness of INCA, we presented three applications to show how such a tool can lower the barrier to entry into this domain, and facilitate creative and iterative exploration within a large and complex design space.

ACKNOWLEDGMENTS

Much of this work was funded through grants from various organizations, including the National Science Foundation, DARPA, the Army Research Lab, Sun Microsystems and Hewlett-Packard. We also thank Maria da Graça Pimentel and her student Renan Cattelan from the University of Sao Paulo in Brazil for being our test users and adopting this work in their research. More information on INCA can be obtained at http://fce.cc.gatech.edu/ubicomp/projects/inca/. For access to the latest version of the toolkit, contact Khai Truong.

REFERENCES

1. Abowd, G., et al. Teaching and Learning as Multimedia Authoring: The Classroom 2000 Project. In the *Proceedings of ACM MM '96* Boston, MA.
2. Brotherton, J.A., Enriching Everyday Activities through the Automated Capture and Access of Live Experiences - eClass: Building, Observing and Understanding the Impact of Capture and Access in an Educational Domain, in College of Computing. 2001, Ph.D. Thesis. Georgia Institute of Technology: Atlanta,GA.
3. Bush, V., As We May Think, in Atlantic Monthly. 1945.
4. Chiu, P., et al. A genetic algorithm for video segmentation and summarization. In the *Proceedings of IEEE Intl. Conf. on Multimedia and Expo (ICME 2000)*.
5. Chiu, P., et al. NoteLook: Taking Notes in Meetings with Digital Video and Ink. In the *Proceedings of ACM Multimedia 1999* Orlando, FL.
6. Cruz, G. and Hill, R. Capturing and Playing Multimedia Events with STREAMS. In the *Proceedings of ACM Multimedia 1994* San Francisco, CA.
7. Davis, R.C., et al. NotePals: Lightweight Note Sharing by the Group, for the Group. In the *Proceedings of CHI 1999* Pittsburgh, PA.

8. Deitz, P. and Yerazunis, W. Real-Time Audio Buffering for Telephone Applications. In the *Proceedings of UIST'01* Orlando, Florida.
9. Dey, A.K., Providing Architectural Support for Building Context-Aware Applications, Ph.D. Thesis. College of Computing. 2000, Georgia Institute of Technology: Atlanta.
10. Dey, A.K. and Abowd, G.D., *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.* HCI, 2001. **16**(1-3).
11. Dey, A.K., et al. The Conference Assistant: Combining Context-Awareness with Wearable Computing. In the *Proceedings of ISWC 1999* San Francisco, CA.
12. Dourish, P., et al. Using Properties for Uniform Interaction in the Presto Document System. In the *Proceedings of UIST 1999* Asheville, NC.
13. Franklin, D., Bradshaw, S., and Hammond, K. Jabberwocky: You don't have to be a rocket scientist to change slides for a hydrogen combustion lecture. In the *Proceedings of IUI '00*.
14. Freeman, E.T., The Lifestreams Software Architecture, Ph.D. Thesis. Department of Computer Science. 1997, Yale University.
15. Guzdial, M., *Supporting Learners as Users.* The Journal of Computer Documentation, 1999. **23**(2): p. 3-13.
16. Hindus, D. and Schmandt, C. Ubiquitous Audio: Capturing Spontaneous Collaboration. In the *Proceedings of Computer Supported Collaborative Work 1992* Toronto, Canada.
17. Mankoff, J.C., An architecture and interaction techniques for handling ambiguity in recognition-based input. Ph.D. Thesis. College of Computing. 2001, Georgia Institute of Technology: Atlanta, Georgia.
18. Mankoff, J.C., Hudson, S.E., and Abowd, G.D. Interaction techniques for ambiguity resolution in recognition-based interfaces. In the *Proceedings of UIST '00* San Diego, CA.
19. Minneman, S., et al. A confederation of tools for capturing and acessing collaborative activity. In the *Proceedings of ACM Multimedia 1995* San Francisco, CA.
20. Moran, T.P., et al. "I'll Get That off the Audio": A Case Study of Salvaging Multimedia Meeting Records. In the *Proceedings of CHI 1997* Atlanta, GA.
21. Mukhopadhyay, S. and Smith, B. Passive Capture and Structuring of Lectures. In the *Proceedings of ACM Multimedia 1999* Orlando, FL.
22. Pedersen, E.R., et al. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. In the *Proceedings of ACM INTERCHI 1993* Amsterdam, The Netherlands.
23. Pimentel, M., Abowd, G., and Ishiguro, Y. Linking by Interacting: a Paradigm for Authoring Hypertext. In the *Proceedings of Hypertext'2000* San Antonio, TX: ACM.
24. Rekimoto, J. Time-Machine Computing: A time-cenric approach for the information environment. In the *Proceedings of UIST '99* Asheville, NC: ACM Press.
25. Rhodes, B.J., Just-In-Time Information Retrieval, in Media Laboratory. 2000, MIT.
26. Richter, H., et al. Integrating Meeting Capture within a Collaborative Team Environment. In the *Proceedings of UbiComp 2001* Atlanta, GA.
27. Shirmohammadi, S., Ding, L., and Georganas, N., *An Approach for Recording Multimedia Collaborative Sessions: Design and Implementation*. Journal of Multimedia Tools and Applications, 2001.
28. Streitz, N.A., et al. DOLPHIN: Integrated Meeting Support across Liveboards, Local and Remote Desktop Environments. In the *Proceedings of Computer Supported Collaborative Work 1994* Chapel Hill, NC.
29. Truong, K.N., and Abowd, G. Personal Audio Loop: Reminders from a PAL. Submitted to *UbiComp 2002*.
30. Truong, K.N., Abowd, G., and Pimentel, M. Vicariously Sharing Captured Web Experiences through an Automated Recommendation System. Submitted to *CSCW '02*.
31. Truong, K.N., Abowd, G.D., and Brotherton, J.A. Who, What, When, Where, How: Design Issues of Capture & Access Applications. In the *Proceedings of UBICOMP 2001* Atlanta, GA.
32. Truong, K.N. and Chiu, P. SpaceTime Browser: A Tool for Interacting with Documents Using Space and Time Attributes. Submitted to *UIST '02* Paris, France.
33. Weber, K. and Poon, A. Marquee: A tool for real-time video logging. In the *Proceedings of CHI 1994* Boston, MA.
34. Wilcox, L., Schilit, B.N., and Sawhney, N. Dynomite: A Dynamically Organized Ink and Audio Notebook. In the *Proceedings of CHI 1997* Atlanta, GA.