The Ektara Architecture: The Right Framework for Context-Aware Wearable and Ubiquitous Computing Applications.

Richard W. DeVaul, Alex "Sandy" Pentland
The Media Laboratory, Massachusetts Institute of Technology
{rich,sandy}@media.mit.edu

Abstract

In this paper we describe the Ektara architecture, a distributed computing architecture for building context-aware ubiquitous and wearable computing applications (UWC). We begin by describing the critical requirements for developing real context-aware UWC applications and relate these to a plausible user-centered scenario. We then present the functional components of the Ektara architecture and explain how they address the critical requirements. Examples of how these functional components interact to create real applications are given, and we discuss our progress in implementing a prototype system and several applications.

1. Introduction

In recent years, a wide range of context-aware wearable and ubiquitous computing applications have been proposed, ranging from location annotation systems to room-based sensing systems to wearable computer-vision systems that facilitate real-space live-action gaming[10, 14, 12, 13]. At the same time, recent work integrating wearable and ubiquitous computing shows the promise of combining these technologies to create applications which leverage the inherent advantages of both[11]. Our review of this work has led us to identify critical features of context-aware wearable and ubiquitous computing systems and to propose a common functional architecture for the development of real-world applications in this domain. The critical, required features include:

1. centralized management of competing demands for the user's attention
2. decentralized contextual resource discovery and allocation
3. a uniform, decentralized mechanism for contextual information storage and retrieval
4. flexible context sensing and classification based on heterogeneous sensors
5. strong cryptography for authentication, privacy, and commerce
6. Open standards for the seamless integration of wearable and ubiquitous computing resources

In addition, we strongly believe in a design philosophy which puts the user and the user's needs at the center of the design process. We developed the Ektara architecture to address the critical features and needs, and to make it easy to build user-centered ubiquitous and wearable computing (UWC) applications.

1.1. Scenario: Alice's Day Off

In order to make a case for the Ektara architecture, we must make a case for the requirements outlined above. We begin with a scenario illustrating the use of UWC systems:

> Alice, an architect, is relaxing on her day off. It is a warm spring day, and she is looking forward to an afternoon of recreation and running a few errands. While she was sleeping, Alice's UWC system (her wearable and the computing resources in her home) was taking care of the mundane tasks of keeping track of Alice's messages, scanning the news and financial services, and managing the climate control of her home. Since this is Alice's day off, her UWC system waits until she sits up in bed to raise the lights in the bedroom and start the coffee brewing in the kitchen.
>
> As Alice gets up, her wearable, which is a translucent plastic lozenge about the size of a deck of cards, glows softly to let her know that there are messages, but none are urgent. She leaves it on the night-stand and enters the bathroom to take a shower.

By the time Alice returns, the wearable is glowing more brightly; her boyfriend called, and the UWC has decided that this message is more important than the others. Alice slips on some clothes, slides the wearable onto her belt, and walks into the kitchen. By now the coffee is ready, and as Alice sits down at the table with her mug, the kitchen's wall display comes to life. It lists three email messages by subject and one voice message by sender (Alice's boyfriend) which is at the top of the list. The headlines of Alice's personalized news and financial reports are shown as lower-priority items. Alice touches the wearable and says "voice messages," and gets up to fix breakfast.

The room's audio system plays Bob's message as Alice looks through the refrigerator. Bob is inviting Alice to meet him at the park that afternoon. Alice pours some cold cereal, notices that she is low on soy milk, and walks into the living room. As Alice sits down at the couch with her cereal, the living room's wall display comes to life with the information that was previously on the kitchen display. "Reminder," Alice says. The wall display changes to show a list of to-do items. "Next time I'm near a grocery store, remind me to buy soy milk." A new reminder is added to the list. The text reads: "Proximity reminder, location grocery store, action message: buy soy milk — confirm?" "Confirmed," Alice says.

Later that day as Alice is walking to the park, she runs into Charles, an old client of hers. After talking for a little while, Alice's wearable alerts her that there is a reminder associated with this situation by displaying an icon on her integrated glasses display. Without pausing in the conversation, she touches the wearable to request the information. The text "swimming pool" appears in place of the icon. "Oh," Alice says, "Did you ever resolve the dispute with your contractor about the swimming pool?"

As Alice walks home that evening she passes through Garibaldi Square, near a Wordsworth store. One of Alice's shopping applications notices that Le Corbusier's "Towards a New Architecture," a book she is looking for, is listed as being on sale in Wordsworth for a lower price than it can be found on the net. She stops in the store and picks up the book. While she is there her wearable automatically discovers an available wall display and she uses it to call up a list of other books she is interested in to check the store's prices. Alice purchases her books by authorizing her shopping agent to transfer cybercash funds to the store, and walks out with a friendly wave to the cashier.

Walking through Alice's day, we learn that:

1. Centralized management of competing demands for the user's attention is critical for any real-world UWC applications because ad-hoc management by individual applications doesn't scale. Pasco notes that one of the features that context-aware systems can provide is contextual adaptation, *i.e.* adjusting the application's behavior to the user's context[9]. However, requiring Alice's cell-phone application to know when she does not want to be interrupted would mean requiring all of Alice's other messaging and reminder applications to know this as well (and they would still compete with each other). The only feasible alternative is a top-level attention management system analogous to the window-management systems employed by conventional GUIs.

2. Dynamic decentralized contextual resource discovery and allocation enables Alice's wearable to discover and use the available resources in any environment, whether it is her home or the bookstore. Kortuem *et al.* describe a centralized mechanism for local resource discovery[7]. However, we believe that the contextual resource discovery process (also described by Pasco[9]) must be dynamic and decentralized to match the unpredictable and constantly changing demands that users place on these resources. Decentralization makes the contextual resource discovery process robust, scalable, flexible, and privacy preserving; Alice can reveal as much or as little about herself as she chooses in requesting access to Wordsworth's UWC system, and the store can then decide whether to grant access on a case-by-case basis.

3. Strong cryptography for authentication, privacy and commerce makes it possible for Alice to trust that as she interacts with a wide range of other UWC resources her personal information and financial transactions will remain private. Rhodes *et al.* have noted the security advantages wearable computing can provide as a means of storing and controlling access to private information[11]. However, if the wearable is unable to transmit or receive this information securely over untrusted networks and unable to authenticate communications with remote hosts, users, and agents, it destroys many of the promised rewards of the integrated wearable/ubiquitous computing environment.

4. A uniform, decentralized mechanism for contextual information storage and retrieval greatly reduces the

number of services required to support a broad variety of context-aware applications. This system must solve three basic problems. First, it must be possible to associated arbitrary documents with a well-structured context description; the stick-e note system described by Brown *et al*. (which provides a fairly general SGML DTD-based context description and querying framework) appears to solve this problem well. Second, it must be possible to associate a well-constructed context description with contextual information servers, so that servers can operate within well-defined contextual scopes. Third, it must be possible for applications to discover servers which match a particular contextual scope (which returns us to the distributed contextual resource discovery problem).

5. Flexible context sensing and classification based on heterogenous sensors enables a broader range of context aware applications than any single sensor or classifier type can support. It has already been shown that computer vision can be used to extract useful location and action context from a wearable's environment[1, 13]. By supporting this type of general-purpose sensing, it is possible to extract a much wider range of contextual information from the user's action and environment than can be obtained with specialized location sensing; it is impossible for Alice's system to recognize "action contexts" such as the "talking to Charles" context, without a flexible sensing and classification system.

6. Open standards for the seamless integration of wearable and ubiquitous computing resources mean that Alice knows her wearable will be able to talk to ubiquitous computing resources everywhere she goes. Further, Alice's wearable can now be a minimalistic core interacting with a range of on and off-body interfaces, computational resources, and networked services.

2. The Ektara Architecture Components

We designed the Ektara architecture to address the needs outlined above. In this section we describe the architecture's functional components and then examine how the components would work together to make Alice's scenario a reality:

2.1. Open-Standards Distributed Computing Foundation

The foundation of the Ektara architecture is an open-standards based distributed computing environment. This is less a component than a requirement for building the rest of the system, but it is important enough to be mentioned here.

2.2. Context-Aware Interaction Manager

The context-aware interaction manager (CAIM) provides a uniform framework for interaction between applications and the user. The primary goal of the CAIM is to minimize the demands on the user's time and attention while maximizing the relevance of the information provided. The CAIM works to achieve this goal by taking into account the HCI resources currently available to the user, important contextual factors effecting the user's ability to pay attention to the UWC system, and the actions of the user's applications. Further, the criteria by which the CAIM makes these decisions must be understandable and controllable by the user; ideally the CAIM should implicitly learn the user's preferences over time yet still be able to provide the user with an explicit description of its decision model. The CAIM's decision model becomes part of the user's personal profile which resides on the wearable and is always accessible wherever she goes.

2.3. Dynamic Decentralized Resource Discovery

The dynamic decentralized resource discovery framework allows UWC applications and services to find and use resources that match semantic descriptions of functionality and context. The foundation of this system is a protocol by which a UWC component obtains networking services and contacts a directory registration service. The UCW component provides the registration service a semantic description of itself and its capabilities, and any additional contextual information it chooses to provide. The registration service then makes further determinations about the resource's context and hands this information off to an appropriate contextual information server (CIS). If this resource becomes unavailable, the registration service informs the CIS and the registration information is removed.

2.4. Contextual Information Service

A stick-e notes like context framework for documents is combined with the idea of a *distinguished* context for servers and a distributed contextual resource discovery mechanism to address the needs for a general-purpose distributed contextual information service (CIS). A distinguished context is a context template which specifies the scope of a contextual message server. For instance, Alice's house runs a contextual message server with a distinguished context matching the "Alice's house" context. The distinguished context mechanism allows resource and personal

information to be stored and accessed where it is most relevant and appropriate. Hence, the private information associated with Alice's home is stored on a contextual information server having that location (Alice's house) as its distinguished context. Likewise, personal information about Alice herself "lives" on a server on Alice's wearable which has Alice's identity as its distinguished context.

The CIS is a distributed database service which provides UWC applications and services a uniform means of storing and retrieving contextual information. Clients may query a server for all records matching a context template or subscribe to receive records when matching information is posted or expires. CIS servers may register the distinguished contexts of other servers, allowing clients to discover other members of the CIS federation.

Like the stick-e notes system, the CIS must support a range of context classifications, including locale, authorship, intended recipient of information, time of posting, time of relevance, time of expiration, deliverability (whether a record is ordinarily intended to be delivered once or multiple times), mime document type, and an extensible mechanism to allow the uniform handling of unanticipated or idiosyncratic contexts.

As in stick-e notes, some contextual qualities may be hierarchical. For instance, Real, physical locations tend to be named in a hierarchical way: a location in a room in a building on a street in a town, *etc.*. The CIS should treat nominal locations context in an analogous way; we propose the convention `<fine-location>/<room>/<building>/<region>/` *etc.*. for describing nominal indoor locations. Likewise, a locale such as a room or building may be described by specifying the asterisk (`*`) wild-card character to match all sub-locales, *e.g.* `*/borglab/E15/MIT` would match all fine-locations in the Borglab room in building E15 on the MIT campus. Likewise, `*/*/E15/MIT` would refer to any location within the Wiesner Building. The CIS also supports a general alias mechanism for nominal contexts, so that the strings `Borglab, borglab,` and `BorgLab` could all be aliases for the "official" room name `384F`.

2.5. Perceptual Context Engine

The perceptual context engine (PCE) is a means of turning raw sensor data and other sources of information into symbolic context descriptions, such as "talking to Charles." The PCE has a two-layer structure, with a perceptual context classifier system at the foundation and an inference system at the top.

The perceptual context classifier system is a signal processing system which converts the raw sensor data into a collection of probabilistic estimates. Conceptually, the classifier system allows the user to train event recognition functions, or classifiers, to recognize patterns in the sensory data and tag them as specific events. The mechanism by which this time-series recognition occurs is a multi-level HMM grammar which is capable of recognizing patterns at a range of time-scales from seconds to days.

The inference system's job is to take the output of the classification system (and other sources of context such as a system clock or GPS receiver) and convert this information into symbolic context descriptions. The inference system also allows for multiple interpretations of the underlying data, such as continuous (latitude and longitude) as well as nominal or discrete (Wordsworth, Garibaldi Square) representations of location.

Although the inference system should support complex inference models (Bayesian inference, graphical models, *etc.*[5, 6]) driven by the classifier data, simple thresholding may be sufficient for many applications if the underlying classifier system is capable enough.

2.6. Strong Cryptographic Security and Authentication

It is a practical impossibility for CIS servers to verify the context of their clients (which are typically executing on untrusted hosts, communicating across untrusted network). Since context-spoofing is trivial, security and authentication in the UCW environment must be achieved through a decentralized public-key cryptography infrastructure. By encrypting the information stored in CIS servers, users can ensure only the intended recipient of that information will be able to decode it even though it is accessible by everyone. Likewise, decentralized distribution of public keys (a "web of trust") and cryptographic signatures allow the verification of authorship and identity, and encrypted messaging allows for the transfer of money and private information over untrusted networks.

3. Ektara Examples

The following examples provide more detail about how the various components of the Ektara architecture might work together in real applications.

3.1. Alice's Morning

When Alice puts on her integrated eyeglasses display it announces itself to her wearable's registration service and provides a semantic description of itself (this might be abbreviated as "HCI device, output, visual, color, small, head-mounted"). The registration service makes the additional context determination that the display is being worn by Alice (location: Alice) and hands this information to the contextual information server (CIS) on Alice's wearable. The

"Alice" location matches the CIS server's current list of aliases for local (Alice is always local unless she takes off the wearable) so the CIS server notifies the context-aware interaction manager (CAIM) service which has subscribed to context events of type "resource: local HCI." The CAIM service decides that the glasses display now provides the best means of visualizing Alice's unhandled messages. The wearable's "chassis glow" notifier light dims off and small message icons appear in Alice's peripheral vision.

As Alice walks into the kitchen, the CIS server receives a new location description ("middle:kitchen:home") from the nominal location inference agent. The wearable's CIS server updates its list of aliases for "local" and re-runs the CAIM's query. The new list of local HCI resources includes the wall display ("HCI host, output, visual, color, large, public"). The CAIM decides that the wall display is now the most appropriate way to display Alice's messages and requests access to it. The wall display, a separate UWC host, checks the cryptographic signature on the request, authenticates Alice's wearable, and grants access. The icons fade from Alice's glasses and message summaries appear on the large screen. Since there is now more visual real-estate available, The CAIM displays lower-priority information (Alice's news and financial headlines) below the message summaries.

3.2. Talking to Charles

A brief description of the classifier system is provided below; for more information see the Vismod technical report TR-519 available online at `http://vismod.www.media.mit.edu/tech-reports/TR-519/`.

The perceptual context engine (PCE) which drives the majority of Alice's UWC applications is sophisticated enough to recognize individual people, given sufficient training. Starting with an "off the shelf" people-finder classifier and a simple training application, Alice has trained classifiers to reliably recognize her coworkers, friends, and a few of her clients.

The classifier takes two inputs, the sensor data from camera and microphone, and the label stream from the user or software agents. The goal of the classifier is to extract meaningful features from the sensor data and use these features to detect the events that the user has labeled. The classifier is based on work done by Clarkson [3, 2], who describes his preliminary classifier system as follows:

1. Extract basic features from the sensors at approximately 5Hz. We calculate all spatial moments up to order 2 from the images, 10 equally spaced frequency coefficients from 50Hz to 8000Hz from the audio, including measurements of auditory volume and the amount of speech detected in the environment.

2. These features are collected continually as the user goes through his/her day of activities. All of them together are used to build a World Model by training a Hidden Markov Model (HMM) with the above features. The resulting World Model is really a rough description of the user's surrounding sensory dynamics.

3. Next as the user labels various events and contexts around him/her with the equivalent of a clicker trainer (i.e. impulse labels that don't specify duration), Event Models are built by training more HMMs on the feature sequences surrounding each of the impulse labels.

4. The resulting Event Models are compared with the World Model to recognize these events after the training phase. $L(Event\ Model|Observations\ at\ t) > L(World\ Model|Observations\ at\ t)$ indicates a triggering of the event detector (where $L()$ indicates the log likelihood function). Or, equivalently we can define an activation function for each classifier as $A(t) = L(Event\ Model|Observations\ at\ t) - L(World|Observations\ at\ t)$.

The output of each classifier is fed into a simple inference agent that acts as a comparator with hysteresis. When the value of the classifier goes high while the uncertainty is low the context event "talking to X" (where X is the name Alice assigned to the inference agent) is created. Alice's proactive reminder application, which subscribes to all of the PCE inference agents, then queries the CIS server and hands the results off to the interaction manager.

Weeks ago, Alice requested the reminder message "swimming pool" to be triggered by a conversation with Charles. This afternoon her PCE correctly identified the chance encounter, the reminder application queried the CIS server, and the interaction manager delivered it.

4. Implementation

Having developed the conceptual framework of the Ektara architecture, we are now in the process of implementing it. In this section, we discuss our progress and then describe some research applications based upon the Ektara architecture.

We chose Hive[8] to prototype the Ektara architecture over other options (including Sun's Jini, custom Java and RMI code, and C/C++ library solutions) for several reasons. First, Hive has proven itself a successful framework for building distributed wearable and ubiquitous computing applications, as has been discussed at length in [11]. Second, Hive's pure Java implementation and available source-code it both portable and hackable. Third, Hive provides

all of the foundation features (except integrated strong encryption) "out-of-the-box," including asynchronous messaging, the distinction between trusted and untrusted resources, and a syntactic and semantic description framework. Finally, Hive provides a built-in distributed resource discovery mechanism and and a robust agents-oriented programming environment.

We have a working prototype of the contextual information service implemented in Hive. The main feature lacking in the present version is the ability of clients to "subscribe" to the service and receive updates whenever the subscription query produces new results. This service, along with Hive's native resource discovery mechanism, provides the beginnings of the dynamic decentralized resource discovery system. Currently applications may search for resources through these two separate mechanisms; a finished dynamic decentralized resource discovery system will require their integration into a single framework.

The perceptual context engine is also in development. Clarkson[3] has implemented a working prototype of a low-level perceptual context classifier, which currently exists as a pure C++ windows application with a sockets interface. Using only a camera and microphone as inputs, the prototype system has been able to achieve a better than %85 accuracy (better than %90 in all cases but one) in recognizing and distinguishing between a group of six actions, such as entering and leaving the wearer's office, *etc.*[3]. In addition, we have developed a skeleton inference system under Hive which does classifier thresholding and produces symbolic context events. We are currently in the process of combining these components into a unified perceptual context engine.

The Context-aware interaction manager is the most complex component of the Ektara architecture (next to the distributed programming system itself), and a clean implementation requires the implementation of all of the other components. The system we envision is based on an "interaction daemon" which asynchronously accepts and processes messages and interaction requests from client applications. The messages produced by a CAIM client contain not only the information to be displayed but also an assessment of how important the information is, what type of user-response is expected, and meta-content that allows the CAIM system to rendering that information in whatever way best suits the demands of context, resources, and application. We hope to implement a working prototype of the CAIM by October 2000.

4.1. Applications

The Ektara architecture is a recent development we have only just begun to implement the components and applications to test them. The first applications were designed to test individual components, such at the contextual information service. One of these applications, Auto-session, allows the wearable user to sit down at any desktop machine and automatically have a secure login session, both to that machine and through that machine back to the wearable. The auto-session application uses an IR-beacon based localization system[4, 12] and employs the CIS to discover Auto-session enabled desk-top HCI resources. Since native strong encryption is still missing in Hive, Auto-session employs SSH key-based authentication to securely validate the user's access to the desktop machine without the need for typed passwords.

Another Ektara test application allows simple "post-it-note" annotation of arbitrary context (typically location) through HTML documents, and then proactively queries the server based on the user's current context; the results are displayed on a web-browser. The distributed Ektara architecture allows multiple HTML-dispatching clients to "listen" to the same query agent, allowing the results of the context queries to be displayed simultaneously by the user's wearable and by other displays in the environment (such as a host with audio capabilities that the wearable lacks, or a large display). This framework is sufficient to provide a simple proactive context-driven reminder application.

The large-scale test application we are now working on is a sophisticated context-aware reminder system very similar to the one described in Alice's scenario. This system will utilize a prototype of the CAIM service (under development) rather than a web-browser to interact with the user and will be driven by a PCE engine capable of recognizing a range of action (as well as location and time) contexts.

5. Conclusions

We strongly believe that the requirements outlined in Section 1 are important considerations for the development of real UWC applications, and that Ektara is the correct architectural response to these requirements. Our implementation results are preliminary, but sufficiently advanced to convince us of the appropriateness and technical feasibility of this approach. It has been said of programming language design that a good language makes the common things easy and hard things possible. We believe that the Ektara architecture does this for creating context aware ubiquitous and wearable computing applications.

6. Bibliography

References

[1] H. Aoki, B. Schiele, and A. Pentland. Realtime personal positioning system for a wearable computer. In *Digest of*

Papers. Third International Symposium on Wearable Computers, pages 37–43. IEEE Computer Society, 1999.

[2] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. In *ICASSP'99*, 1999.

[3] B. P. Clarkson. Recognizing user's context from wearable sensors. Technical Report 519, Vision and Modeling Group, MIT Media Lab, January 2000.

[4] A. R. Golding and N. Lesh. Indoor navigation using a diverse set of cheap, weearable sensors. In *Digest of Papers. Third International Symposium on Wearable Computers*, pages 29–36. IEEE Computer Society, 1999.

[5] F. V. Jensen. *An Introductino to Bayesian Networks*. Springer Verlag, New York, 1996.

[6] M. I. Jordan, editor. *Learning in Graphical Models*. Kluwer Academic Press, 1988.

[7] G. Kortuem, Z. Segall, and M. Bauer. Context-aware, adaptive wearable computers as remote interfaces to intelligent environments. In *Digest of Papers. Second International Symposium on Wearable Computers*, pages 58–65. IEEE, October 1998.

[8] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: Distributed agents for networking things. In *Proceedings of ASA/MA'99, the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents*, 1999.

[9] J. Pascoe. Adding generic contextual capabilities to wearable computers. In *Digest of Papers. Second International Symposium on Wearable Computers*, pages 92–99. IEEE, October 1998.

[10] A. Pentland. Smart rooms, smart desks, smart clothes: Toward seamlessly networked living. MIT Media Lab Vision and Modeling Group Technical Plan, January 1996.

[11] B. J. Rhodes, N. Minar, and J. Weaver. Wearable computing meets ubiquitous computing: reaping the best of both world. In *Digest of Papers. Third International Symposium on Wearable Computers*, pages 141–149. IEEE Computer Society, 1999.

[12] T. Starner, D. Kirsh, and S. Assefa. The locust swarm: An environmentally-powered, networkless location and messaging system. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 169–170. IEEE Computer Society, 1997.

[13] T. Starner, B. Schiele, and A. Pentland. Visual context awareness in wearable computing. In *Digest of Papers. Second International Symposium on Wearable Computers*, pages 50–57. IEEE Computer Society, October 1998.

[14] C. R. Wren and A. P. Pentland. Dynamic models of human motion. In *Proceedings of FG'98*, Nara, Japan, April 1998. IEEE.