Computation for the Corridor: Experiments in Augmenting Public Spaces

Max Van Kleek (emax@csail.mit.edu)

December 15, 2004

1 Introduction

The OK-net project was conceived in 2002 to design interactive digital information displays for high-traffic public spaces in the workplace. The aim of the project was to build systems that naturally complemented these spaces and the activities that took place therein, in ways that had not yet been explored by existing digital information displays. We saw opportunities for three different types of OK-net applications: applications that acquired and delivered relevant content to passers-by, applications that turned the displays into two-way channels of informal social communications among members of an organization, and applications that, in real time, supported the face-to-face conversations and interactions that occurred near these displays. We hoped that the enhanced information dissemination capabilities for informal communications provided by such applications would lead to greater social awareness and connectedness within organizations.

In the process of designing these applications, we realized the need for a large number of capabilities not afforded directly by any off-the-shelf platform toolkits. Thus, we embarked to first design a software architecture that embodied these capabilities. This archiecture, the *OK-net software platform* extended the AIRE Metaglue agent framework to facilitate display coordination, information persistence, and sharing. We also designed the platform to make it easier to integrate emerging systems that provided new, perceptual ways of interacting with users, such as through vision, gesture, and speech. We believed these interaction technologies would play a key part in making OK-net displays useful, effective and attractive to users.

At the end of the first phase of the project, we finished laying the initial architectural groundwork for the OK-net software platform, and implemented an initial suite of applications. This platform and applications were then installed on a number of OK-net information kiosks that were deployed in four locations within our laboratory. We then extended the platform to incorporate speech-based interaction, and performed a user study evaluating users' impressions. This was followed by an examination of how to best identify returning users for the purposes of display personalization, for which a gesture-based authentication scheme was designed and evaluated.

Although the applications currently running on the kiosks still have limited functionality and perceptual capability, they have already received widespread use and generated substantial user feedback. This feedback will permit the building of the next generation of applications, for which designs are currently underway. This paper describes the OK-net project from conception and motivation through experimentation, implementation, and initial deployment. It discusses several contributions to ubiquitous computing and HCI, and concludes with a discussion of what challenges remain at the close of the first phase of this project.

2 Background

2.1 The role of public spaces in the workplace

Public areas within the workplace are settings for day-to-day social activities that may not always be associated with work, but which nonetheless have been found to contribute to the well-being of individual workers, as well as to the overall health of the organization. Paper bulletin boards, located on walls in public lounges, hallways, or foyers, have remained important, even as e-mail and the web have become accessible nearly everywhere. These conventional bulletin boards help to build an organizational sense of community, by serving as valuable social and informal communications channels outside of the normal work context. The settings for traditional bulletin boards are also locations where workers most frequently chat with colleagues who may hail organizationally from outside their immediate workgroups. The resulting short, unplanned informal meetings have been shown to play an important role in forming collaborations, making social connections, and discovering and exchanging expertise. The increased recognition of impromptu exchanges in public spaces has led architects and workplace designers to re-think the allocation of space within the workplace. From the office-block layouts that maximized private office space in the 1960's (visible in the notorious 'cubicle farms' of today), offices have evolved toward open, common areas emphasizing shared space. Yet despite these changes, little information technology has come into common use in these public spaces; hence, their usefulness has remained limited. changed little compared with the modern the modern office desk.

2.2 Knowledge workers' need to collaborate

As predicted originally by economist Peter Drucker in 1959, the *knowledge worker*, a new class of skilled worker whose job was exclusively to create, manipulate, and disseminate information, came to be a dominant figure in the work force of the United States by the end of the 20th century [7]. As time went by, knowledge workers not only required a high level of specialization, but also often needed to accomplish tasks that required experience outside their immediate realms of expertise. While collaboration with others with the appropriate expertise would at least allow them to accomplish these tasks more efficiently, it was often difficult for workers to find and form appropriate collaborations outside their immediate work-groups. The aforementioned public spaces in the workplace comprised places where random, chance social encounters were likely to occur among people with a broad diversity of expertise and background. Thus, these spaces increasingly were seen as settings for social interactions that formed valuable *weak ties* for all individuals within an organization.

According to Mark Granovetter's sociological theory of weak ties, individuals who created an extensive network of casual acquaintances became hubs for communication between groups of close friends, and thus played a key role in the dissemination of ideas and knowledge within organizations. Granovetter supported his theory by analyzing how the social well-connectedness of organizations correlated with their effectiveness at responding to crises or market volatility. This theory also demonstrated that, on average, a greater proportion of one's opportunities, such as for job placements and new collaborations, arose more through one's weak ties than from one's closest friends and colleagues. [9] Thus, Granovetter implied that improving the level of social connectedness would have positive implications for both individuals and their organizations as a whole.

2.3 Workplace Surveys

Findings from a number of recently conducted workplace studies have suggested that random encounters among coworkers in the workplace often have led to informal, unplanned meetings, and that these meetings made important contributions to daily work-related activities. These findings

The Steelcase Workplace Index Survey, entitled "Employees On the Move", conducted by Opinion Research Corporation (ORC) in May 2002, studied 977 office workers in a variety of settings in order to determine what workplace offerings enhanced the quality of people's lives the most. The findings were unexpected, particularly for Steelcase, a company specializing in the design of ergonomic desks and chairs. Figures published by the study indicated that, on average, less than half of the participants' work-week was spent sitting at their desks. The remainder of the time was spent largely at meetings away from their desks, holding impromptu meetings "in secondary spaces, such as hallways, enclaves, and at water coolers" [19].

Further evidence supporting the transition from formal meetings to informal gatherings was revealed by a study for the iLAND project led by Norbert Streitz of the German National Research Center for Science and Technology (GMD). This study interviewed 80 members of creative industrial design teams on how technology currently played a role in their design meetings, compared with how they would like technology to optimally shape their meetings in the future. The study revealed that most team members felt that formal "brainstorming" sessions in meeting rooms were run archaically, rarely utilizing the available computer technology in the creative process. Instead, team members felt that meeting rooms of the future should "have the character of a marketplace, or a landscape providing opportunities for spontaneous encounters and informal communication" and that "team meetings [should not be] conducted by meeting in a room, but by providing an environment and a situation where encounters happened." [20].

3 The OK-net platform

3.1 Design Objectives

The objectives of OK-net were twofold: first, to design an open platform for perceptually-enabled computers in public spaces such as hallways, lounges, and the like; and secondly, to design a number of applications, on top of this platform, that could provide a variety of useful information and services. For the former, we included scenarios for applications which would require limited perceptual or inferential capabilties not necessarily present in desktop applications today. In particular, we wished to design applications to disseminate information that would enable people to be more socially aware of one another and their immediate organizational surroundings.

Since there were no open pre-existing systems for public spaces that met our needs, we decided to design OK-net on top of commodity desktop PC software and hardware. This created a number of design problems at a variety of different levels. At the hardware level, we had to determine how existing, commodity desktop personal computers could be modified into a form-factor that would be suitable for these spaces, and that could be reasonably obtained and installed. At the level of the human-computer interface, we had to determine what interaction technologies would be most appropriate for these spaces. This included determining whether and how the system could interact with users' personal devices, such as laptops or mobile phones. At a software level, we had to determine how to aggregate and represent information, and ensure a level of autonomy and reliability. In addition, we had to determine how information retrieval, dissemination, and storage would be coordinated across a number of different OK-net terminals. We also had to consider how the software could integrate novel human-computer interaction technologies, such as perception and sensing. Finally, and most importantly, at the application level, we wanted to identify what types of information and capabilities would be most interesting or useful to people in these spaces, and how best to deliver these services to them. The remainder of this paper will focus on the software aspects of the design problem; for details on OK-net hardware design, please see **??**.

3.2 Related Work

Our design goals crossed a number of disciplines and research areas in computer science, sociology, and human-computer interaction. In this section, we will discuss relevant work from research in computer-human interaction, artificial intelligence and computer supported cooperative work (CSCW) communities.

3.2.1 Social awareness

As workspaces became increasingly physically fragmented, improving the social connectedness of individual workers became correspondingly important.

Since the advent of the *Portholes* project at EuroPARC in 1992 [6], a number of research systems demonstrated various approaches for supporting awareness in distributed workgroups both on and off the desktop. One approach was to integrate awareness affordances into existing desktop tools, such as groupware [21] and e-mail and instant messaging (IM) clients. Applications that supported informal awareness off the desktop included instant messengers for mobile phones[22], media spaces on large shared displays in public spaces [3], and digital bulletin boards within workgroup spaces [10]. Another approach taken by a large number of research projects was to use *ambient displays* to provide subtle peripheral social awareness within the physical environment. [12]

3.2.2 Knowledge management and social software

Knowledge management (KM) tools were designed to facilitate the formation of collaborations among members of large organizations, particularly large corporate enterprises, in part, by identifying who possessed particular expertise on various topics. A central KM server within the enterprise determined this by performing statistical textual analyses on databases of documents produced by individuals, identifying keywords that might indicate knowledge in particular areas. When someone needed consultation on a particular topic, the KM database would provide a list of people who were likely to be able to help.

More recent work, such as Henry Kautz's ReferralWeb, built models of social networks, inferring the types and strengths of interpersonal relationships among members of an organization, to evaluate potential candidates for collaboration. Kautz theorized that people would feel more comfortable initiating a collaboration with someone with whom they were at least nominally connected, (e.g., a mutual friend) rather than with a complete stranger. [13]

3.2.3 Supporting impromptu meetings

Tools to support unplanned, impromptu meetings remained relatively uncommon, compared with those designed for meeting rooms and offices. Among the few commercial products designed to support informal meetings included the Huddleboard from Steelcase, a lightweight portable whiteboard created for unplanned meetings in random public spaces in the workplace. [2] DUMMBO, a research prototype from Georgia Tech, resembled an ordinary whiteboard, but was instrumented to capture all marker strokes and audio, within a limited range, to a circular buffer which was accessible for download immediately from any desktop. [4] The BlueBoard, an augmented touchscreen plasma display from IBM Research's BlueSpace project, was similar to OK-net in many respects, and was intended to allow workers to collaboratively compare their personal calendars and to exchange documents. Users accessed their personal documents by authenticating themselves to the system using their corporate-issued RFID cards[17].

3.2.4 Perceptual interaction techniques

OK-net integrated a number of prototype systems to collectively enable more natural computer-human interaction, including vision-based user detection, speech recognition, and speech synthesis. Thus, OK-net could be viewed as an experiment to determine how well these systems could work with one another.

One system that served as an inspiration for the project included the Digital Research Smart Kiosks project, which integrated a vision-based face tracker and speech input/output into an information kiosk form factor, incorporating an expressive animated avatar which users could interact with while requesting information or playing a game.

4 Software Architecture

4.1 The need for a common, knowledge-rich data representation

The design process for the OK-net software architecture was initiated with the building of several applications for informal gathering spaces. The first such application was *k:info*, (to be described in section 5.1), a dynamic digital billboard designed for large displays in high-traffic spaces that cycled through a variety of news and current-events. To minimize the amount of maintenance required, *k:info* was designed to automatically update its own content. Specifically, it would periodically retrieve content from a variety of information sources, such as e-mail, the web, and RSS feeds. It would then display them, eliminating them when they became obsolete. In many cases, these accumulated documents were descriptions of persons, places, or things. Most often, they comprised information about events that would happen in the near future. Instead of storing the documents themselves for display, it became clear that it was more natural and more generally useful to build and store representations of the things described. For example, by parsing an email about a talk announcement into a representation of the event itself, k:info could use logistical information such as where and when the event would take place, and what topics would be discussed, to determine how, how often, and to whom to display it.

The need for a knowledge representation became even more evident when we undertook to make k:info context-aware. Instead of displaying items according to some fixed and pre-determined schedule (as with most present-day digital information displays), k:info followed rules that specified what types of content would be useful to display under different situations. These fuzzy rules, or *heuristics*, selected items based on conditions such as the current time of day, weather conditions, or the number of people nearby. In order to express and represent these variable conditions, it became clear that a language, or *ontology* was needed. Furthermore, it required the system to be able to interpret the specified heuristics within the context of what articles were available, in order to infer a choice of what was appropriate to display. This demonstrated the need for an *inference engine* that could interpret statements and to make deductions from prior knowledge.

Thus, k:info inspired the need to choose a knowledge representation and inference framework which could be used for both traditional system data (such as information about users and items to display), as well as observations about the world that could be used for enabling applications to perform context-aware decision-making. The next section describes Metaglue's knowledge representation and storage facilities, and introduces Ontogen, a simple language for expressing application ontologies for knowledge stored in Metaglue's built-in semantic network representation. Following that is a discussion of ContextKeeper, a forward-chaining blackboard approach that was used in an initial implementation of k:info. Lessons learned from ContextKeeper led us to a third approach, a distributed query architecture for opportunistically answering queries about the world.

4.2 Background: Knowledge representation in Metaglue

Prior to the Metaglue Semantic package, there was no common representation for knowledge in the Metaglue system. Knowledge about the world was held in a purely distributed fashion among individual agents, and agents would contact and communicate knowledge with one another on an ad-hoc basis. This was initially viewed as an advantage, since it meant that there was no need to have an explicit agreed-upon common ontology among agents. In practice, however, this yielded several difficulties. One problem was that when an agent needed a piece of knowledge, there was no clear mechanism to figure out whom among the myriad of other agents to consult.

The solution originally proposed for this problem was to reference the Resource Manager (RM) as an intermediary, who would take a request (in the form of the name of an interface), determine "the most appropriate agent" to handle the particular request, and return a concrete reference to the chosen agent. In practice, however, this made building a resource manager impractical, as it placed too much responsibility on the RM. Essentially, the fact that the RM was required to make judgments about the suitability of all the other agents in order to respond to a particular request meant that the RM had to have expertise about all other agents in the system. Then, the RM was required to measure the efficacy of each, (which not only was a highly subjective measure, but moreover was usually application- and request-specific). Being able to judge the suitability of a particular agent to handle a request also meant that the RM had to predict how agents would respond, since the outcome might affect an agent's suitability (such as if an agent had no knowledge or response). This, of course, no resource manager could ever do reliably, since the task of predicting an outcome of an algorithm was computationally undecidable. As a result, resource managers were rarely used for this purpose in practice, causing agent writers to resort to hand-coding agent identifiers and agent-agent relations at compile time.

As will be discussed in 4.4, knowledge-centric approaches such as blackboard architectures eliminated this problem by allowing whatever agents were available to act opportunistically by themselves to contribute to a problem solving effort, thus enabling effective loose coordination among agents.

4.2.1 The Metaglue Semantic package

The Metaglue Semantic package, introduced by Peters [16] as part of his Hyperglue extensions to Metaglue, provided a framework for building semantic networks out of instances of Java classes that represented individual nodes and links. While the serialized triples-representation of the Metaglue semantic network strongly resembled RDF, there were a number of differences. Nodes and links were strongly typed, because they directly corresponded to instances of Java classes derived from the package's Node or Link class. As a result, both metaglue semantic nodes and links could have fields. Once created, semantic network elements were made persistent by serializing themselves into triples stored in any jdbc-compatible SQL database.

4.3 A simple ontology language for OK-net

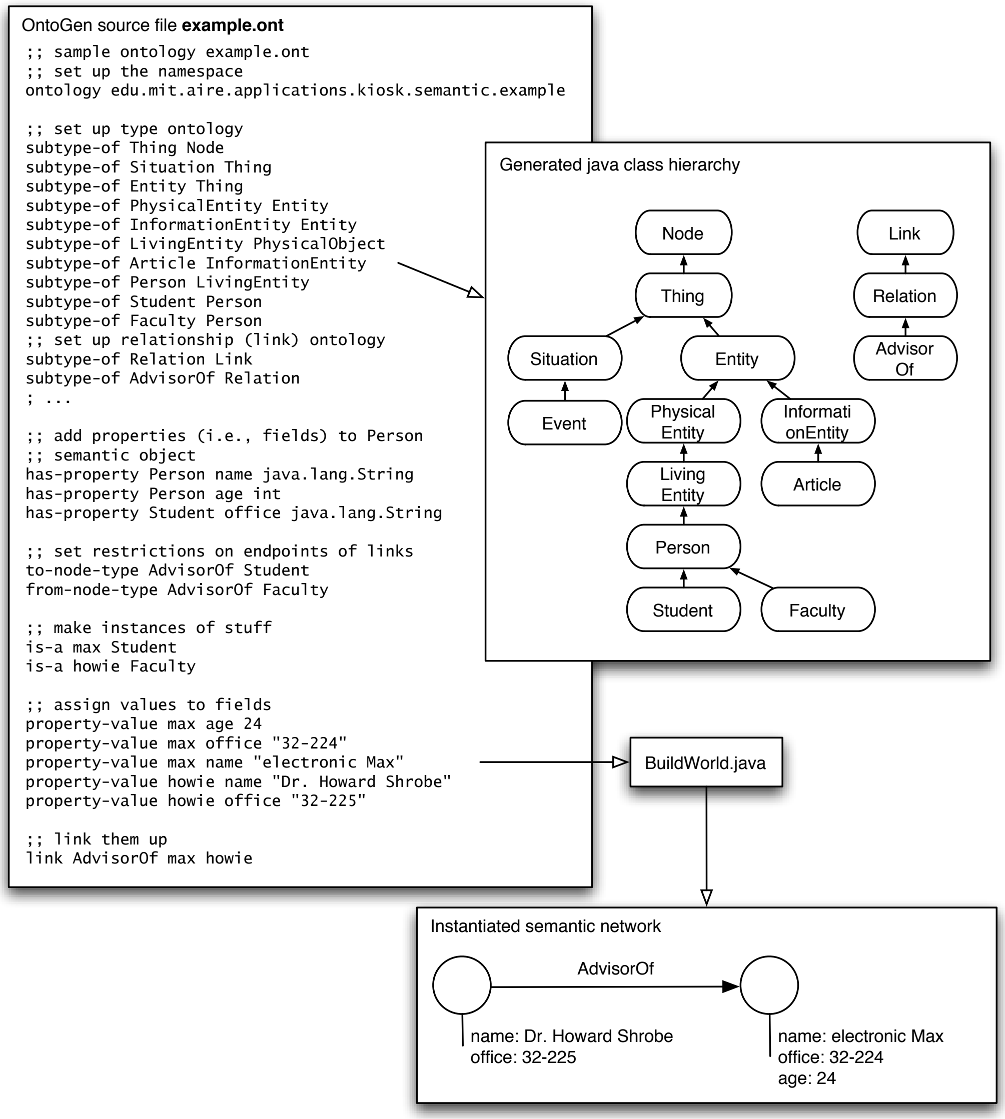
To facilitate defining new ontologies for metaglue applications, OK-net introduced *Ontogen*, an ontology definition language. When definitions of derived node and link types defined in the Ontogen language were compiled, they were transformed into the appropriate Java source files corresponding to each of the derived types. Instances could also be expressed in ontogen, which corresponded to assertions about the world. These assertions were added to a program known as BuildWorld, which, when executed, filled the active semantic DB with all instance knowledge in the ontology definition. An example of the ontogen description language is illustrated in figure 1.

Ontogen for Metaglue corresponded to the Web Ontology Language, or OWL, for the Semantic Web. While the expressive power of OWL as a description logic vastly exceeded that of Ontogen, the simplicity of Ontogen allowed its entities to be directly mapped onto Java types, which made objects in the semantic network easy to manipulate from a software engineering standpoint. Mapping ontology relationships onto the Java type hierarchy system also eliminated the need for a reasoner to compute types at runtime. Specifically, the expressiveness of Ontogen was approximately equivalent to that of RDFS, sans support for multiple subclass relationships (due to Java's single-inheritance restriction).

4.4 Version 1: A blackboard architecture for context inference

The first context inference framework designed to fulfill k:info's needs was a *blackboard architecture* known as ContextKeeper. ContextKeeper was inspired by Terry Winograd's recent work with Interactive Workspaces [27] which applied a traditional AI problem-solving architecture to the problem of context inference. In contrast to a traditional remote procedure-call (RPC) or context widget ([5]) model, the blackboard approach, he contended, promoted a knowledge-centric, rather than process-centric approach. Additionally, blackboards enabled coordinated problem solving among agents, by defining roles for each of the various agents, and a protocol for inference within the task of context resolution.

ContextKeeper was designed as a centralized respository where Metaglue agents could store their own assertions about what they wished to share about what they knew about the world. All such assertions were expressed in terms of types defined in a common ontology type hierarchy (or T-Box) to which all agents had access. Each assertion carried an identifier indicating from which


OntoGen source file **example.ont**

```
;; sample ontology example.ont
;; set up the namespace
ontology edu.mit.aire.applications.kiosk.semantic.example

;; set up type ontology
subtype-of Thing Node
subtype-of Situation Thing
subtype-of Entity Thing
subtype-of PhysicalEntity Entity
subtype-of InformationEntity Entity
subtype-of LivingEntity PhysicalObject
subtype-of Article InformationEntity
subtype-of Person LivingEntity
subtype-of Student Person
subtype-of Faculty Person
;; set up relationship (link) ontology
subtype-of Relation Link
subtype-of AdvisorOf Relation
; ...

;; add properties (i.e., fields) to Person
;; semantic object
has-property Person name java.lang.String
has-property Person age int
has-property Student office java.lang.String

;; set restrictions on endpoints of links
to-node-type AdvisorOf Student
from-node-type AdvisorOf Faculty

;; make instances of stuff
is-a max Student
is-a howie Faculty

;; assign values to fields
property-value max age 24
property-value max office "32-224"
property-value max name "electronic Max"
property-value howie name "Dr. Howard Shrobe"
property-value howie office "32-225"

;; link them up
link AdvisorOf max howie
```

Figure 1: *Defining ontologies in Ontogen*. Type descriptions in ontogen are transformed into separate java classes that descend from *Node* and *Link*. Descriptions of instances are written to a Java class called *BuildWorld*, which, when executed, generate a semantic network in the relevant Hyperglue semantic database.

agent it came, allowing agents using that information to directly query the source of information. Metaglue agents could subscribe to particular types of assertions and be informed by a callback mechanism when these assertions were added. Agents that contributed assertions to the system were known as *Knowledge sources*. One general purpose knowledge source was a JESS [1] rule-chainer agent that could be fed rules expressed in a CLIPS-like rule syntax, and would forward-chain over assertions on the blackboard and assert conclusions back to the blackboard.

4.5 Version 2: DBCBB: A distributed query architecture for context

Implementing ContextKeeper revealed that a forward-chaining blackboard was not as suitable for a distributed agent architecture as originally imagined. First, from a philosophical standpoint, it seemed counter to the nature of Metaglue to force agents to consolidate the knowledge that they had kept in a distributed fashion among agents, back into a single database. From a system design perspective, there were the disadvantages of creating a potential scalability bottleneck as well as a potential single point of failure. A greater problem, however, stemmed from the volatile nature of the data being modeled. Specifically, the problem derived from the fact that agents which acted as knowledge sources to the blackboard could not determine whether the new knowledge that it received would be useful to other agents or to the application. Therefore, agents were continually obliged to report changes about the world. Since features of context that could potentially be useful to applications were often highly dynamic (frequently changing aspects of the world, such as fluctating temperatures or proximity readings from sensors measuring distances to users nearby), updates were necessitated extremely frequently, causing constant cascades of inference.

An additional issue came from the problem of truth maintenance. Since facts pertaining to context inference attempted to model a dynamic system (ie., the world), assertions true at one moment in time might not be true at each successive moment. In a forward chaining framework, this knowledge would have to be explicitly retracted, set to expire, or somehow made to be superseded by successive contributions. The framework would then have to eliminate the entailments that were no longer true. Since in ContextKeeper, assertions were constantly being retracted, truth-maintenance became extremely expensive.

To eliminate these problems our second system, the *Distributed Back-Chaining Black Board* or *DBCBB*, took a backward chaining approach for making applications contextually aware. In this approach, an application needing to know about the world issued a high-level query to its underlying inference framework. The framework routed the query to various knowledge sources which attempted either to solve the query with knowledge they had, or to break the query down into smaller subqueries which had a better chance of being solved, prior to passing them on. Agents with the necessary evidence to satisfy a query produced that evidence, which then was unified with the original high-level query and returned to the application. By being *application-driven* rather than *context-* or *data-driven*, this approach allowed knowledge sources to produce information on demand, whenever the application needed it. Furthermore, keeping knowledge local to agents until it was needed eliminated the constant updating of an external, centralized knowledge repository. Answers to queries were interpreted as an agent's proposed answer, true at the particular moment the query was satisfied instead of being a presentiment about the future. As a result, there was no need for any truth maintenance mechanism.

The design of DBCBB started with a traditional Scheme pattern-matching backwards chaining rule engine, which was integrated in Metaglue using the Kawa Scheme interpreter in Java**??**. Queries assumed the form of tuples of symbols, possibly containing wildcard variables. When a query was made, the agent first attempted to satisfy the query itself using its own internal knowledge base and rules. If no solution were found, it dispatched the query to all other query-capable agents in the user's society. Each of these agents, in turn, attempted to resolve the query with its own rules and knowledge bases, until a solution was found. Since some agents had rules that "translated" a particular query from high-level queries to lower-level (ie. more concrete) ones, a single query could result in a backchaining inference cascade among the agents, enabling the distributed problem solving behavior with localized knowledge. The inference query propagation mechanism eliminated inference loops among agents by passing the current *goal stack* along during each successive query invocation, and by comparing each rule's antecedents against the goal stack prior to attempting each rule. Informally, if an element in the antecedent matched an element in the goal stack, this implied that the rule required currently unknown knowledge, and thus the rule would not provide any more information.

In the Metaglue implementation of DBCBB, agents wishing to issue or answer queries extended from QueryableAgent, a new type of metaglue *ManagedAgent* derived for this purpose. Each QueryableAgent had an independent instance of an inference engine and its own local knowledge base. This provided some flexibility by allowing multiple agents to have differing "opinions", or beliefs of what was true about the world, even among agents within the same society. It was left to the application to determine which of the responses to heed if it received multiple answers. The application could decide it needed to explicitly call the agents that helped resolve the original query in order to provide more information to clarify any ambiguities.

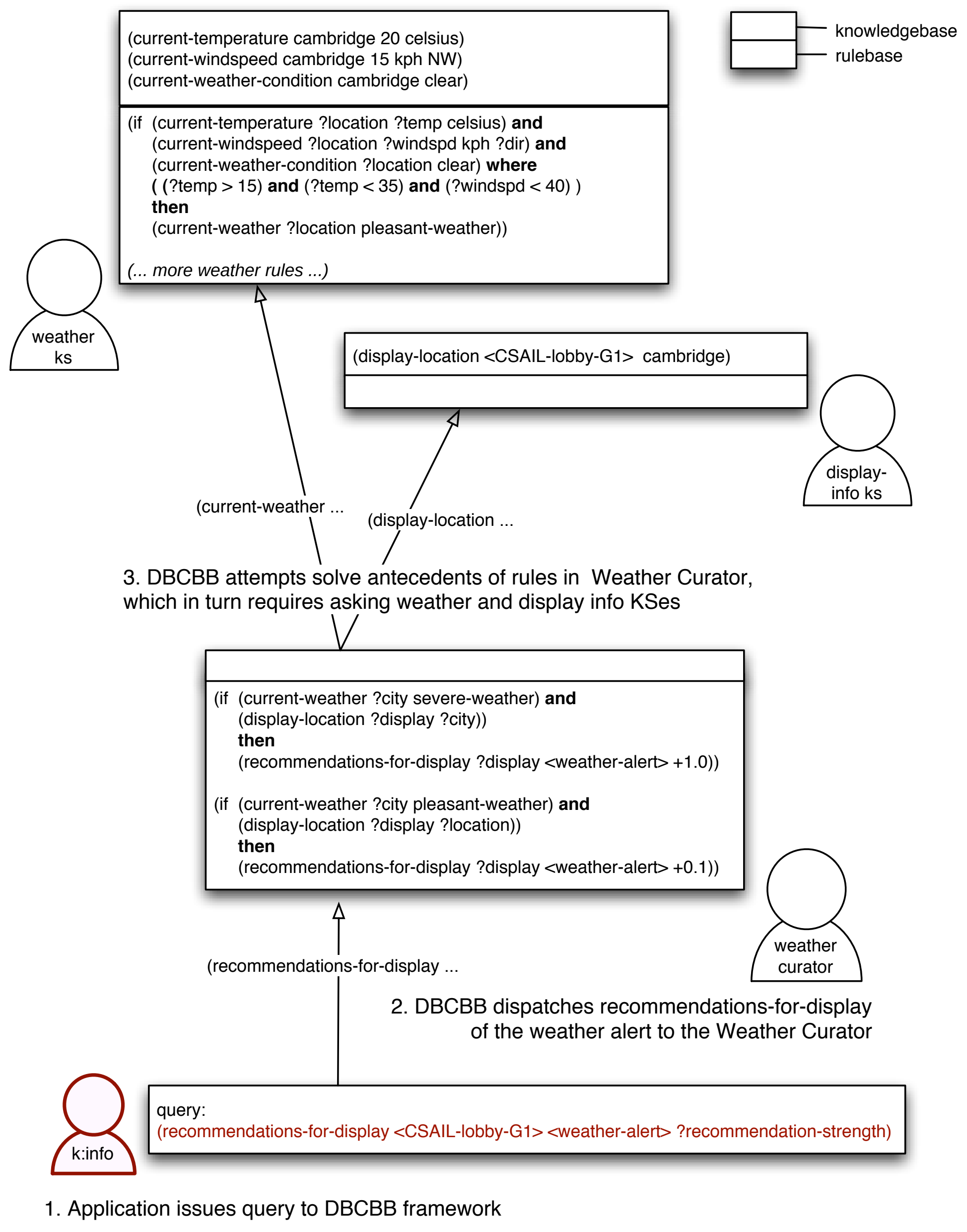
5 Applications

5.1 k:info: a "Smart" Billboard

As described earlier, *k:info* was conceived as a dynamic information billboard application that automatically updated itself from a variety of news and information sources, and curated itself based upon what it perceived that people nearby were likely to want to see. The Metaglue agents that managed content for display fell into two categories: knowledge sources and curators. Knowledge sources were responsible for pulling in relevant content for the display. Examples of knowledge sources currently in use at CSAIL included agents that retrieved content from e-mail mailing lists, XML RSS feeds, static web pages, and web services. The behavior of the display (ie., what k:info chose to display at any particular moment) was governed by the curators, whose job it was to nominate particular items for display. To choose among candidates, the curators themselves might rely upon other knowledge sources, referred to as *contextual knowledge sources*. An example of a contextual knowledge source would be one that inferred nearby users' interests, based on, for example, previous interactions with a kiosk.

The k:info application called for candidate items and asked curators to vote for their favorites. Votes, assigned arbitrary integer values, were tallied for each candidate element, and the elements with the largest number of votes won selection. In the event of a tie, k:info made its selection among the tied candidates randomly.

In practice, we found that breaking up behaviors into individual curators allowed us to yield complex behaviors from a relatively simple set of agents. For example, the Least Recently Displayed curator, was designed to ensure variety by keeping track of when each piece of content was last presented, and voting for those which had not been displayed in a while. In conjunction with



knowledgebase
rulebase
(current-temperature cambridge 20 celsius)
(current-windspeed cambridge 15 kph NW)
(current-weather-condition cambridge clear)
(if (current-temperature ?location ?temp celsius) **and**
(current-windspeed ?location ?windspd kph ?dir) **and**
(current-weather-condition ?location clear) **where**
((?temp > 15) **and** (?temp < 35) **and** (?windspd < 40))
then
(current-weather ?location pleasant-weather))
(... more weather rules ...)
weather ks
(display-location <CSAIL-lobby-G1> cambridge)
display-info ks
(current-weather ...
(display-location ...
3. DBCBB attempts solve antecedents of rules in Weather Curator, which in turn requires asking weather and display info KSes
(if (current-weather ?city severe-weather) **and**
(display-location ?display ?city))
then
(recommendations-for-display ?display <weather-alert> +1.0))
(if (current-weather ?city pleasant-weather) **and**
(display-location ?display ?location))
then
(recommendations-for-display ?display <weather-alert> +0.1))
weather curator
(recommendations-for-display ...
2. DBCBB dispatches recommendations-for-display of the weather alert to the Weather Curator
k:info
query:
(recommendations-for-display <CSAIL-lobby-G1> <weather-alert> ?recommendation-strength)
1. Application issues query to DBCBB framework

Figure 2: Illustration of how k:info, the dynamic billboard application, uses DBCBB to determine what to display by soliciting votes from agents for candidate content items. In this figure, k:info is trying to determine whether to display the weather report next, which causes a *curator agent* to need to assess the actual weather conditions near the display. To obtain this information, the weather curator queries the DBCBB, which connects it to yet another agent, which is directly connected to a web-based weather service.

other curators such as the Severe Weather curator, which would vote for displaying weather more frequently during unpleasant weather conditions, the resulting behavior was one that generally made sense to end users, i.e., interposing severe weather warnings frequently within other news content. In the future, we may consider implementing a subsumption architecture **??** in place of the simple voting scheme, so that higher priority curators, such as an Emergency Message curator, could proactively suppress certain lower-priority curators.

5.2 SKINNI: The Stata Center Information Guide

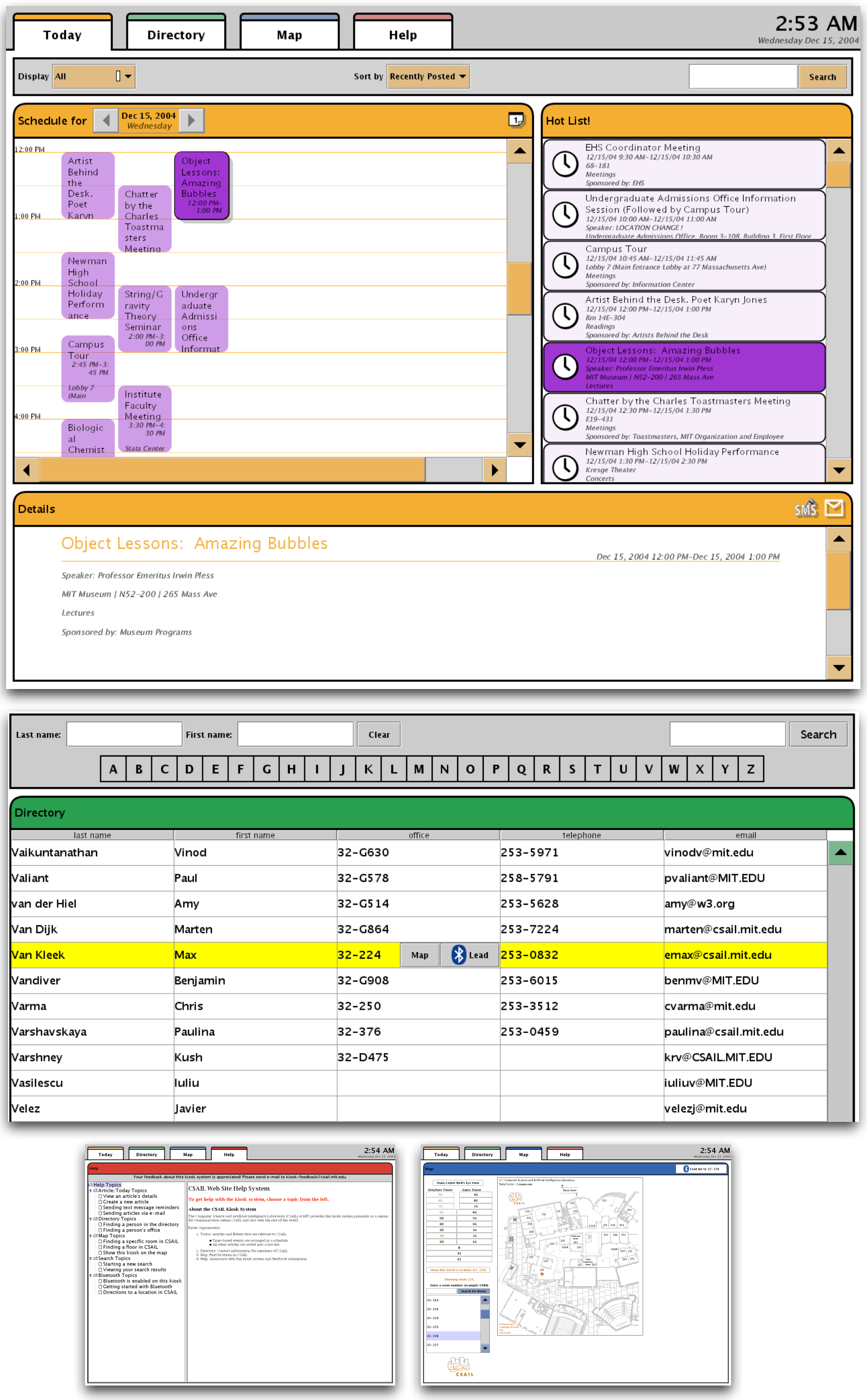
The Smart Kiosk Information Navigation and Noteposting Interface (SKINNI) for OK-net was an interactive graphical user interface for touch-screen displays that allowed users to explicitly look up a variety of lab-related information, including current events provided by a variety of k:info knowledge sources as described earlier, coupled with a lab directory and map information. The original motivation for SKINNI for our laboratory came from two primary observations of working at CSAIL. First, it was observed that lab members often had trouble keeping track of the large number of seminars, invited talks and special events that took place each day in the laboratory. Students, in particular, reported that they frequently forgot about talks they had wished to attend, even after receiving e-mail reminders that were sent by the organizers. Secondly, the lab seemed to lack a cohesive sense of community among lab members, particularly among members from different research groups, or whose offices were on different floors of the building. A preliminary informal survey of 10 CSAIL lab members supported these observations: the percentage of people they could identify from among research groups that were situated on floors other than their own within the building was generally under 10%, compared to an approximately 75% for members who had offices on the same floor. [23] Our goal, thus, was to raise lab members' general awareness of both social and research-related events, as well as members' awareness of one another.

SKINNI presented the day's events in a format similar to a number of popular calendaring applications such as Apple's *iCal* or Microsoft's *Outlook*, but had a number of unique features. First, as already discussed, content was updated automatically by the same agents that managed content for k:info. Second, SKINNI could send timely reminders of events as text messages to users' cell phones shortly before the events transpired.

5.2.1 *Lead me*: a Bluetooth-based building tour guide for SKINNI

To demonstrate how multiple OK-net terminals could work together in tandem to enhance SKINNI's functionality, we developed an extension to SKINNI known as *Lead me*. When a user queried SKINNI about a location, such as looking up a location on SKINNI's map, or looking up the location of a lab member's office via the directory, and the system detected that the user might be equipped with a bluetooth[1] -enabled device (that supported the OBEX push-protocol), it offered to lead the user to the destination. If the user chose this feature, SKINNI would contact the rest of the OK-net kiosks about the user's desire to get to a particular destination. Then, as the user walked around the building passing by OK-net kiosks in elevator lobbies, he/she would receive updated instructions and maps of how to get to the desired destination. This feature would terminate when either the user reached the closest node to the destination, or after 10 minutes.

[1]Bluetooth is a standardized short-range wireless peripheral communications protocol that we chose due to its recent popularity and support by portable digital device and mobile phone manufacturers. See http://www.bluetooth.com for further details.



Artist Behind the Desk. Poet Karyn

Chatter by the Charles Toastmasters Meeting

Object Lessons: Amazing Bubbles 12:00 PM-1:00 PM

Newman High School Holiday Performance

String/Gravity Theory Seminar 2:00 PM-3:00 PM

Undergraduate Admissions Office Informat

Campus Tour 2:45 PM-3:45 PM Lobby 7 (Main

Institute Faculty Meeting 3:30 PM-4:30 PM Stata Center

Biological Chemist

EHS Coordinator Meeting
12/15/04 9:30 AM–12/15/04 10:30 AM
68–181
Meetings
Sponsored by: EHS

Undergraduate Admissions Office Information Session (Followed by Campus Tour)
12/15/04 10:00 AM–12/15/04 11:00 AM
Speaker: LOCATION CHANGE !
Undergraduate Admissions Office, Room 3–108, Building 3, First Floor

Campus Tour
12/15/04 10:45 AM–12/15/04 11:45 AM
Lobby 7 (Main Entrance Lobby at 77 Massachusetts Ave)
Meetings
Sponsored by: Information Center

Artist Behind the Desk. Poet Karyn Jones
12/15/04 12:00 PM–12/15/04 1:00 PM
Rm 14E–304
Readings
Sponsored by: Artists Behind the Desk

Object Lessons: Amazing Bubbles
12/15/04 12:00 PM–12/15/04 1:00 PM
Speaker: Professor Emeritus Irwin Pless
MIT Museum | N52–200 | 265 Mass Ave
Lectures

Chatter by the Charles Toastmasters Meeting
12/15/04 12:30 PM–12/15/04 1:30 PM
E19–431
Meetings
Sponsored by: Toastmasters, MIT Organization and Employee

Newman High School Holiday Performance
12/15/04 1:30 PM–12/15/04 2:30 PM
Kresge Theater
Concerts

Object Lessons: Amazing Bubbles
Dec 15, 2004 12:00 PM–Dec 15, 2004 1:00 PM

Speaker: Professor Emeritus Irwin Pless

MIT Museum | N52–200 | 265 Mass Ave

Lectures

Sponsored by: Museum Programs

last name	first name	office	telephone	email
Vaikuntanathan	Vinod	32–G630	253–5971	vinodv@mit.edu
Valiant	Paul	32–G578	258–5791	pvaliant@MIT.EDU
van der Hiel	Amy	32–G514	253–5628	amy@w3.org
Van Dijk	Marten	32–G864	253–7224	marten@csail.mit.edu
Van Kleek	Max	32–224	253–0832	emax@csail.mit.edu
Vandiver	Benjamin	32–G908	253–6015	benmv@MIT.EDU
Varma	Chris	32–250	253–3512	cvarma@mit.edu
Varshavskaya	Paulina	32–376	253–0459	paulina@csail.mit.edu
Varshney	Kush	32–D475		krv@CSAIL.MIT.EDU
Vasilescu	Iuliu			iuliuv@MIT.EDU
Velez	Javier			velezj@mit.edu

Figure 3: SKINNI user interface. *Top:* The *today view* allows users to quickly discern the day's events, with timeline and list views of events. *Middle: Directory* view enables quick searching and browsing of the CSAIL directory. *Bottom:* Help and map information.

This feature was very well received when it was demonstrated at the Stata Center, although in practice it was difficult to provide adequate guidance throughout the building due to the limited number of OK-net terminals we had deployed.

5.3 Serendipity: Initiating informal interactions between coworkers

Serendipity was an application that took proactive measures to allow coworkers to get to know one another. Serendipity ran a match-making algorithm that was triggered when the system detected two users within close proximity. Serendipity users ran an application on their mobile phones, which used Bluetooth to detect other users with mobile phones within a radius of approximately 10 meters. Results of scans were compared against a list of registered users on the Serendipity server. The server determined whether to take action and try to inspire an interaction, first by consulting whether it had ever done so for a particular pair of people before, and if not, by gauging whether the potential mutual benefit of the introduction would exceed the users' thresholds for interruptions. This metric, which we referred to as the *potential utility gain* of an introduction, was computed by taking a normalized combination of the intersection of the users' *interest profiles* with a weighted inner product between term frequency keyword vectors extracted from the users' email inboxes. While static profiles bootstrapped the system by providing preferences and interests explicitly, the inner product of the extracted term frequency statistics attempted to model approximately how coincident the users' current interests and expertise were. When the potential utility gain of an interaction exceeded a user-adjustable "proactivity threshold" for both parties, Serendipity attempted to initiate an interaction. When this occurred, it first determined whether or not there was an OK-net kiosk nearby. If a kiosk nearby was available, as illustrated in figure 4, it influenced the k:info billboard running on the display to select items that correlated with the users' mutual interests. The idea was that this would help "break the ice", by providing a topic of conversation that both people could relate to. If no kiosk was nearby, Serendipity resorted to sending MMS text messages to both parties with a short description of the other person, and a little thumbnail picture (from each user's profile).

One of the primary challenges in designing Serendipity was to preserve the users' privacy. In particular, users' interest profiles extracted from their e-mails and other authored documents tended to be extremely sensitive, and users would most likely not want this to be shared with strangers. To solve this problem, we decided to keep user models local to users on their own desktops such that when Hyperglue's access control policies took effect, Hyperglue would ensure that only the agents belonging to the user would be allowed to access or modify this profile. We also attempted to give each user as much control over the profiling process as possible, such as by allowing the user to easily start or stop the profiling process, and to specify a skip-list for e-mails that they did not want the system to see. When the server needed to compare two users' interest vectors for similarity, the agents within the user's society first hashed the keys of the interest vector prior to giving it to the server. This ensured that even if intercepted, the topics themselves would not be easily identifiable, while similarity could be computed just as effectively.

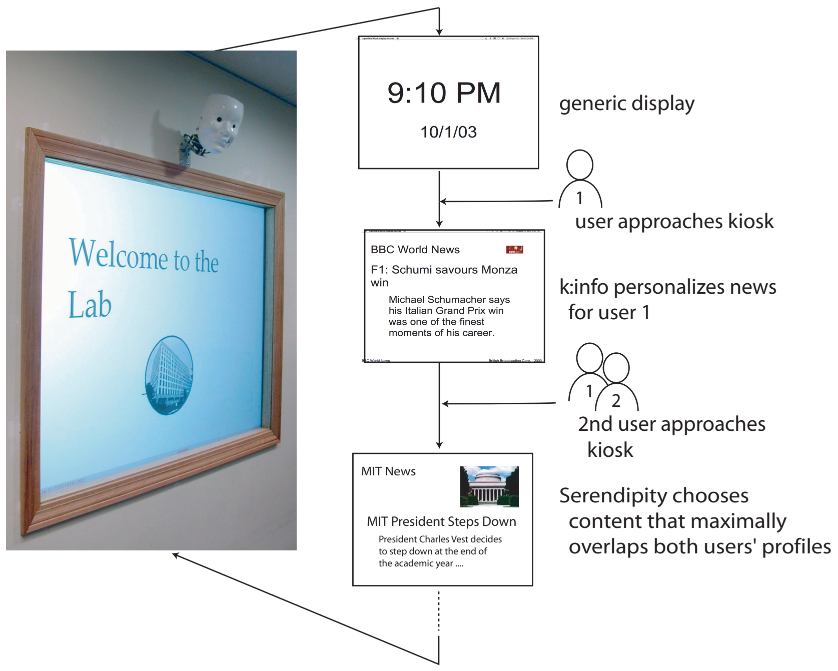

Welcome to the Lab
9:10 PM
10/1/03
generic display
1
user approaches kiosk
BBC World News
F1: Schumi savours Monza win
Michael Schumacher says his Italian Grand Prix win was one of the finest moments of his career.
k:info personalizes news for user 1
1
2
2nd user approaches kiosk
MIT News
MIT President Steps Down
President Charles Vest decides to step down at the end of the academic year ....
Serendipity chooses content that maximally overlaps both users' profiles

Figure 4: Serendipity interaction with k:info. Serendipity acts as a *content curator* for k:info, recommending content that is most similar to both users' interests when Serendipity senses via bluetooth that two users who could mutually benefit from knowing one another are nearby.

6 Interaction technologies: Lightweight identification

6.1 Personalization requires identification

As public information displays became increasingly ubiquitous, active personalization, in turn, became critical, in order to provide positive user experiences and to prevent the displays from being perceived as mere annoyances or distractions. One critical necessity for providing a personalized user experience was for the system to know something about the user. However, since users should not be obliged to trust blindly the myriad public displays they encountered throughout their day with their personal information, public terminals should require as little private information as possible. A compromise was to try to have displays automatically deduce what they needed to know about the users based on observations of their explicit actions. Just as many web sites today took advantage of users' anonymous clickstream data to start tailoring content prior to the users' logging in, public displays did not actually require having the users' specific identities in order to start tailoring the displays. Although it had been shown that building such user models with limited access to personal information yielded correspondingly limited possibilities for personalization, [15] anonymous interaction data could be predictive of various aspects of future usage patterns, such as when the user was likely to visit again [14].

Unlike anonymous clickstream data, which contained distinguishing identifiers that helped to indicate successive visits, such as the user's network (IP) address, interactions at touchscreen displays usually did not have any immediately distinguishable characteristics for returning users. In order to enable the construction of user models that spanned multiple visits, the display required a means by which it could recognize returning visitors.

While a traditional log-in and password approach would suffice in some cases, this approach had a number of drawbacks. First, ensuring one's security would require having a different password and login for each display. Results from a small in-house survey of 15 CSAIL lab members indicated that most were already using between 5 and 10 distinct systems every day, each requiring a username and password; furthermore, they were obliged to memorize between 6-10 unique passwords. While this might seem manageable, as the number of displays increased, users would either be forced to reuse usernames and passwords, which would compromise security, or memorize an unmanageable number of username/password combinations. Secondly, typing one's username and password while standing at a public kiosk terminal, on a potentially unfamiliar keyboard, could also be unreasonably slow compared to the total amount of time spent interacting with the kiosk, a factor which might make users reluctant to log-in. Another in-house survey of 10 users timed how long it took to type in a username and password repeatedly for 6 trials each. While typing times varied wildly, no users were faster than 2.8 seconds, with an average of 6.06 seconds and a standard deviation of 3 seconds across all trials.

Biometric identification techniques, such as voiceprint, face-id or fingerprint, or RFID provided other options. While effective at identifying a user, biometric identification techniques relied on a physical aspect of a user that was (generally) tied to their physiology. Therefore, users' principals, or the digital identity from the displays' perspectives, became inextricably tied to users' actual identities, which resulted in an inappropriate privacy risk for use on untrusted displays. RFIDs required users to either use a tag that they carried around (with its incumbent privacy implications), or to carry a different tag for each system, which had practical limitations.

6.2 *Lightweight* identification

Thus, we proposed new methods of identification for use in building models for personalization on untrusted public systems, known as *lightweight identification* techniques. Unlike traditional authentication techniques which were based on their effectiveness against being compromised, lightweight identification schemes were evaluated with the following criteria:

1. *Anonymity* - The degree to which users' lightweight principals were intrinsically tied to any identifying information about a person's *actual* identity; lightweight identities would not be traceable back to their original owners, and a person would be allowed to obtain multiple principals or change principals whenever they wished.

2. *Ease of use* - The speed and facility with which users could identify themselves to a system; effective lightweight schemes would be able to identify a person within a negligible fraction of total interaction time with the system, and with little cognitive or physical effort.

3. *Size of cognitive footprint* - The nature and amount of information a user must remember in order to identify him- or herself to a system; a smaller footprint eased the task of memorizing mappings from systems to identification keys, and allowed users to more easily keep multiple principals straight, such as for different systems.

4. *Cognitive persistence* - How persistent the information required to identify oneself remained over time.

5. *Scalability* - The maximum size of the set of potential users from which a user could be uniquely distinguished.

6. *Degree of guarantee* - The traditional measure of “security”; the degree of certainty that a person was, in fact, the owner of the principal for which they were identified.

The relative importance of each of these criteria when evaluating a particular scheme depended on the application. For personalizing public displays and information kiosks, anonymity, ease of use, and scalability were likely to be of primary importance. Since public kiosk systems were untrusted, protecting the anonymity of the user on public displays eliminated the potential for “Big Brother” tracking scenarios, and also prevented the accumulation of potentially sensitive information. The second criterion was created for practical reasons; since interaction times with kiosks were typically very short, the time and effort required for identifying oneself to a system must be sufficiently small for the system to be useful. Finally, scalability was very important, since public displays could have an extremely large potential user base. Meanwhile, however, the degree of security (or certainty that a user is who he or she claims to be) was likely to be less important than for an application that, for example, provided access to a user’s sensitive information.

6.2.1 Related research

Much of the inspiration for lightweight identification came from Weinshall and Kirkpatrick’s recent work on utilizing observed human memory phenomena for user identification that could be performed with little apparent conscious effort [25]. In essence, they described a lightweight identification technique based on the imprinting/recognition memory of the human visual perceptual system that was scalable, easy for users to perform, and yielded a high degree of confidence for recognition.

6.3 User identification with Distinctive Touch

Our system, known as *Distinctive Touch*, or DT, was a lightweight identification scheme for touch-screen information kiosks. As described in [24], DT made use of human tactile memory to allow users to easily identify themselves: users created their own “passdoodles”, unique, identifying personal scribble gestures, which they performed on a display. New users demonstrated their gesture five times, based upon which the system would build a model using both doodle shape and gesture velocity. A gesture recognizer then determined the most similar model out of its library of users, and generated a confidence level corresponding to how closely the doodle matched the particular user.

Passdoodles featured two main advantages for public touch-screen displays over the traditional use of textual username and password. First, doodling was easier and faster to perform while standing at a touchscreen display than typing a username and password on either an on-screen or attached physical keyboard. Secondly, recalling and remembering gestures such as passdoodles relied more on one’s implicit motor learning capabilities than one’s conscious semantic memory, resulting in its more easily being memorized, and recalled with less conscious effort. In addition, unlike biometric identification schemes such as fingerprinting or face-id, doodles are anonymous and virtually untraceable back to their owners.

Our initial implementation of DT built simple models of doodle shapes and compared candidate gestures with the model using a minimum description length measure. During an informal

10-user evaluation, we collected 7 positive samples of each user's chosen doodle. We then performed leave-one-out cross-validation, by omitting a sample from each user's set, building models from the remaining samples, and classifying the held out sample. Our recognizer classified 69 out of the 70 collected samples correctly. Furthermore, the time required to perform each passdoodle was consistently under 2 seconds, with the average taking 1.08 seconds. Unfortunately, however, due to the limited number of classes (users), we were unable to gauge the scalability of the algorithm to a large user population. We were also unable to ascertain how much cognitive overhead remembering passdoodles required compared with traditional passwords. However, findings from an earlier user study [8] suggested that the ease of memorization of paper-based passdoodles compared closely to that of chosen textual passwords.

We are currently working on improving our doodle-recognition algorithm to incorporate comparison of gesture velocity over time. This would have the potential of enhanced scalability, by allowing simiarly-shaped but differently executed passdoodles to be distinguished from one another, as well as improved security by making impersonation of passdoodles more difficult. We are planning a larger, follow-up study to evaluate the performance of our new algorithm.

6.4 User identification with mobile devices

An alternative approach reduced user effort by offloading the task of identifying oneself to portable digital devices that users might already be carrying around with them, such as PDAs or mobile phones. Wu, *et. al.*, demonstrated [28] a scheme for securely accessing remote services through potentially insecure public terminals. We tried a number of simpler, lightweight approaches more suitable for personalization that reduced the amount of work required of the user during identification. In the simplest such approach, new users downloaded a small digital token to their cell phone which contained their unique user identification code. When users later returned to the display, they used their cell phone to present this token back at the display. Our implementation used a small application (a midlet) on the mobile phone to make a connection to the display (via the cellular data network) and transmit the user identification code. In practice, the efficiency of this approach proved to be limited by two factors: the speed with which users could locate their mobile phone, and delays experienced in making the data connection. As wristwatches become bluetooth-enabled [11], we may see both of these delays diminish.

7 Interactions using speech

Informal discussions with kiosk users led us to believe that the touchscreen interface was far from ideal for many purposes. In particular, when a user wished to look up a specific piece of information, such as looking someone up in the CSAIL directory, or finding directions to a specific room within the laboratory, using the SKINNI GUI via touchscreen displays seemed tedious. It required an elaborate sequence of button taps or a combination of button taps and keyboard entries to narrow down a search, or even a brute-force visual scan of a long list of items. Even after several GUI design iterations we were unable to come up with a better layout that made GUI navigation more efficient, without increasing the initial learning curve. Therefore, we decided to augment SKINNI, using natural language via speech with the hope that this would improve the overall user experience, through a more direct and natural query interface.

7.0.1 Design considerations

Many of the original design requirements of the SKINNI GUI interface applied to the speech interface as well. The interface required a low usability threshold, ie., it had to be easily accessible for new users. In addition, the interface had to accommodate users of diverse ages, nationalities and computer experience.

Speech interfaces posed a variety of additional usability challenges. Since speech interfaces were still relatively new and uncommon, users were likely to be inherently less familiar with them than with GUIs, and would have widely different expectations of their interaction capabilities. Furthermore, since our speech system was limited to pre-specified forms of queries, rather than unconstrained spoken interaction, it became a challenge for the application designer to ensure that the system could recognize the most common forms of queries. Finally, the user interface had to deal with user misrecognition, since this was likely to occur more often than with a GUI interface.

Acoustic requirements had also to be considered when designing speech interfaces for public kiosks. Users were unlikely to want to put on a headset microphone for short interactions, and far field microphones generally captured too much ambient noise. Directed noise-canceling linear array microphones provided a good compromise at a distance of 2-3 feet; however, they required users' mouths to be positioned within a limited physical area near the center of the array, making the system inaccessible to users of various heights or users in wheelchairs.

7.0.2 Current Implementation

For our implementation, we opted to use the in-house conversational speech engine, Galaxy [18]. Galaxy was attractive for its speaker-independent speech recognition engine that was resilient to outside noise, speech disfluencies, and accents, as well as for its advanced dialogue and discourse capabilities.

At present, we outfitted two prototype kiosks for speech, with speakers and active noise-canceling linear array microphones. A sound daemon running on these machines performed speech detection and capture, and transmitted speech waveforms to a separate machine designated as the Galaxy Hub. The hub coordinated recognition and natural language processing, and passed a parsed utterance back to a SKINNI-speech component known as the *back-end script*. This script interpreted the parsed utterance, sent actions to the SKINNI GUI, computed a verbal response, and updated the speech feedback UI. The response to be verbalized was sent back to the Hub, which passed it to the appropriate text-to-speech (TTS) engine, and sent synthesized waveforms back to the kiosk, where they were played by the sound daemon. The Galaxy domain, consisting of recognition grammars, hub and startup scripts, were created via SpeechBuilder [26], a web-based tool for building Galaxy domains.

7.0.3 Speech state feedback GUI

Initial impressions from using speech on the kiosks led us to realize the importance of conveying immediate feedback about the state of the speech system to the user. Particularly due to the variable delays incurred in processing the speech, parsing the result, and network transmission, without feedback users were left unsure as to whether the system had heard the speech, encountered an error, or was still in the midst of processing. In response, we designed a GUI that displayed immediate feedback about whether speech was enabled (via a red/green microphone icon), when speech was detected, when it was being processed, output from the recognizer, and, finally, a textual rendering of the spoken response from the back-end script. We found that this allowed the user to easily determine when the kiosk was listening, why or how an error occurred, and how to proceed thereafter. This was particularly helpful when new users were familiarizing themselves with the speech interface since it allowed them to quickly grasp the system's capabilities. We believed this would help reduce the number of users who gave up on the speech interface too early if they initially experienced problems efficiently interacting with the kiosks.

7.0.4 Designing the Domain Grammar

The most challenging aspect of building the system was designing the recognition grammar that Galaxy used to interpret spoken utterances. We needed to determine all the basic ways people would phrase their queries. Due to the large number of different phrasings possible for even the simplest queries, we limited our initial speech capabilities to only directory field queries (ie. office, telephone number, email address) and "show me room"-type queries for the map. To determine the most common phrasings of these queries, we initially built up what were considered the most common forms, and then asked 10 lab members how they would phrase the queries. We then extracted common forms from their phrasings and added them to the grammar. In the future, we hope to design a system that can incrementally learn new phrasings for queries.

7.0.5 Speech User Evaluation

To gauge the usability of our system, we performed a preliminary, informal user study with 10 lab members. Our study asked subjects to look up the telephone number of eighteen different people in the lab (selected randomly), using the kiosks. We asked our subjects to use the speech interface for the first six names, the touchscreen interface for the second six names, and their own preferred interface for the final six names.

As indicated in Table 1, our study yielded mixed results. As we wished to evaluate how well first time users would perform, subjects were not told what forms of queries the kiosk could recognize. This was reflected in the recognition rate for the first two trials being particularly low, at 50%. However, the rate improved to 72% thereafter. Misrecognitions caused severe time penalties as a result of requiring an exhaustive search by the speech recognizer, combined with the time required to generate a spoken error response. A combination of the high error-rate and misrecognition penalty yielded a longer average time for speech over touch. However, when the recognition was successful, times were consistently shorter with low variance. Also, it was encouraging that 8 out of our 10 subjects preferred using the speech interface when given a choice between the two.

7.0.6 Current Progress and Future Work

We are currently working on improving the speech domain, as well as updating our voice models to improve recognition accuracy. With help from the Spoken Language Systems group, we are also starting to build new voice models based on raw acoustic data from kiosks rather than models built from telephone data. Furthermore, we are working to extend our existing speech domain such that all functionality exposed via our touchscreen interface is also exposed via the speech interface.

	Unrecognized Speech(secs)	Recognized Speech(secs)	Overall Speech(secs)	Touch (secs)
Min	10	3	3	4
Max	25	9	25	19
Mean	16.78	5.22	9.11	7.33
S.D.	5.13	0.92	6.24	3.18

Table 1: Study Results: The times indicate the interval between when the subject hears the name that needs to be queried and when the subject extracts the corresponding telephone number from the screen.

References

[1] JESS: the Java Expert System Shell. Sandia National Laboratories, 2004.

[2] The Steelcase Huddleboard. http://www.steelcase.com/na/ourcompany.aspx?f=12611&c=15879.

[3] Stefan Agamanolis. icom: a multipoint awareness and communication portal for connecting remote social spaces. http://www.medialabeurope.org/hc/projects/icom.

[4] J. Brotherton, K. Truong, and G. Abowd. Supporting capture and access interfaces for informal and opportunistic meetings, 1999.

[5] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.

[6] Paul Dourish and Sara Bly. Portholes: supporting awareness in a distributed work group. In *Conference proceedings on Human factors in computing systems*, pages 541–547. ACM Press, 1992.

[7] Peter F. Drucker. The age of social transformation. *The Atlantic Monthly*, November 1994.

[8] Joseph Goldberg, Jennifer Hagman, and Vibha Sazawal. Doodling our way to better authentication. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 868–869. ACM Press, 2002.

[9] Mark Granovetter. The strength of weak ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[10] Elaine M. Huang and Elizabeth D. Mynatt. Semi-public displays for small, co-located groups. In *Proceedings of the conference on Human factors in computing systems*, pages 49–56. ACM Press, 2003.

[11] IBM lab demonstrates special bluetooth watch. http://www.mobilemag.com/content/100/342/C1241/, August 2002.

[12] Hiroshi Ishii, Craig Wisneski, Scott Brave, Andrew Dahley, Matt Gorbet, Brygg Ullmer, and Paul Yarin. ambientroom: integrating ambient media with architectural space. In *CHI 98 conference summary on Human factors in computing systems*, pages 173–174. ACM Press, 1998.

[13] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, 1997.

[14] Alan L. Montgomery. Using clickstream data to predict www usage. http://www.andrew.cmu.edu/user/alm3/papers/predicting www usage.pdf, August 1999.

[15] Balaji Padmanabhan, Zhiqiang Zheng, and Steven O. Kimbrough. Personalization from incomplete data: what you don't know can hurt. In *Knowledge Discovery and Data Mining*, pages 154–163, 2001.

[16] Stephen Peters. *Knowledge Representation and Resource Management in Distributed, Pervasive, Intelligent Environments*. PhD thesis, Massachusetts Institute of Technology, 2004.

[17] Daniel M. Russell. Staying connected: Digital jewelry and more. technologies for people. In *Proceedings of SHARE 1998*, 1998.

[18] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. Galaxy-II: A reference architecture for conversational system development, 1998.

[19] Employees on the move. *Steelcase Workplace Index Survey*, April 2002.

[20] Norbert A. Streitz, Jorg Geisler, Torsten Holmer, Shin'ichi Konomi, Christian Muller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-LAND: An interactive landscape for creativity and innovation. In *CHI*, pages 120–127, 1999.

[21] John C. Tang and Monica Rua. Montage: providing teleproximity for distributed groups. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 37–43. ACM Press, 1994.

[22] John C. Tang, Nicole Yankelovich, James Begole, Max Van Kleek, Francis Li, and Janak Bhalodia. Connexus to awarenex: extending awareness to mobile users. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 221–228. ACM Press, 2001.

[23] Max Van Kleek, Tyler Horton, and Elizabeth Boyle. SKINNI: Connecting coworkers using information kiosks in the workplace. Unpublished., 2004.

[24] Christopher Varenhorst. Passdoodles: A lightweight authentication method. http://oknet.csail.mit.edu/papers/varenhorst.pdf, 2004.

[25] Daphna Weinshall and Scott Kirkpatrick. Passwords you'll never forget, but can't recall. In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1399–1402. ACM Press, 2004.

[26] E. Weinstein. Speechbuilder: Facilitating spoken dialogue system development. Master's thesis, MIT, 2001.

[27] Terry Winograd. Architectures for context. *HCI Journal*, 16(4), 2001.

[28] Min Wu, Simson Garfinkel, and Rob C. Miller. Secure web authentication with mobile phones. In *Proceedings of the Student Oxygen Workshop 2003*, Cambridge, MA, 2003.