Intelligent Environments for Informal Public Spaces: the Ki/o Kiosk Platform

by

Max Van Kleek

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2003

Author .....
Department of Electrical Engineering and Computer Science
February 2003

Certified by .....
Howard E. Shrobe
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Intelligent Environments for Informal Public Spaces: the Ki/o Kiosk Platform

by

Max Van Kleek

Submitted to the Department of Electrical Engineering and Computer Science
on February 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

The Ki/o Kiosk Project is a research effort to explore the potential applications for embedding computation in informal public spaces, such as hallways, building lobbies, and break areas. By deploying a prototype kiosk network, the ki/o Project identified challenges, requirements, and design criteria for designing kiosk Intelligent Environments, and fielded community reactions to the system.

Thesis Supervisor: Howard E. Shrobe
Title: Principal Research Scientist

Acknowledgments

This project would have never been possible without Jack Costanza, AI Lab technical manager and local superhero, who provided the inspiration, resources and guidelines for the project. This project also could not have come to fruition without Howie Shrobe, who has provided endless guidance since its beginning.

Special thanks also go to Ron Wiken, a More-Magician who can make miracles take physical form, for leading the physical construction of all the kiosk prototypes.

Thanks also go to Andy Chang and Krzysztof Gajos, who helped me lay the foundations of the project, and to Justin Maynard, Mark Pearrow, Jason Bell and Tyler Horton, for contributing their energy and code to Ki/o. Kimberle Koile and Mark Adler also contributed indispensable guidance and creative insight to the project.

Special thanks also go to my parents and entire family in Idaho for their support, and for reading and reviewing my awful drafts late into the night. Last minute thanks also go to Danny Gomez, for attacking my grammatical bugs with fury, and to Erica Peterson for helping me keep myself together.

Finally, acknowledgements go to Morr Music AG and Warp Records UK for publishing sublime electronic music that propelled me throughout the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is often heard that conversations started "around the office water cooler" lead to some of the most interesting and memorable interactions in today's workplace. This project investigates the role of informal, opportunistic interactions in public areas of the workplace, and explores how ubiquitous computing systems may be built to improve these interactions.

This chapter introduces the research agenda, provides historical context, and presents the motivations behind the work in the remaining sections of this paper.

1.1 What is Ubiquitous Computing?

The field known as *ubiquitous computing* [1] [2] is devoted to changing the relationship between humans and computers towards allowing computers to become invisible and recede into the periphery of people's lives. As the world has become increasingly reliant on personal computers and the Internet, computers have begun to complicate and dominate, rather than simplify everyday tasks. Moreover, computers have come to occupy increasingly more physical space on desks, and in modern living environments, while, at the same time, they consume increasingly

[1] The term "*ubiquitous*" is used interchangeably with the term "*pervasive*" in the research community. This paper adopts the former term.

[2] The origin of the term and the concept is generally attributed to the late Mark Weiser from the Xerox Palo Alto Research Center in the early 1990's.

more amounts of time, require more attention, and demand more mental faculties to run simple tasks. Computer systems today demand that the user be responsible for translating what users want to accomplish into a representation the systems can understand. Much of the exertion required to operate computers originates in having to continuously learn how to properly perform this translation, using whatever clues provided by the system designers. Ubiquitous computing reverses this process, by making computers responsible for the translation from the physical world into the system's representation. Thus, ubiquitous computing systems act like intelligent personal assistants, capable of understanding what people are trying to accomplish in order to determine how best to intervene and assist them. Therefore, ubiquitous computing systems allow people to concentrate on what is truly important, *ie.*, their actual tasks, rather than focusing on the onerous steps of operating the computer systems to perform these tasks.

Ubiquitous computing also eliminates the artificial notion of a personal computer as an independent, isolated computational entity. It instead proposes that computation should be available everywhere as a shared natural resource, just like the air we breathe[39]. Thus, ubiquitous computing is a departure from personal computing, and is widely considered the "third great wave in computing", after the development of time-shared mainframes, and the subsequent rise of personal computers [59].

1.2 What are Intelligent Environments?

Since ubiquitous computing systems need the ability to deduce users' intentions, preferences, and the state of the world automatically, they require the ability to perceive the physical world, interpret these observations, make inferences, and then take appropriate action.

When these systems, capable of perception, cognition, and action, are embodied in a physical space, they are collectively known as *Intelligent Environments*[3], or IEs.

[3]The term *intelligent environment* is often used interchangeably with the terms *smart spaces*,

To date, prototype IEs have been developed primarily for such spaces as conference rooms, classrooms, and offices. The Ki/o Kiosk Platform extends this notion by designing IEs specifically for informal public spaces, such as hallways, lounges, break rooms, and elevator lobbies.

The ultimate vision of ubiquitous computing is that Intelligent Environments will pervade all physical spaces, thereby enabling access to digital information anywhere and at any time.

1.3 The Field of Human-Computer Interaction

To build effective IEs, certain requirements are immediately apparent; it is clear that advances in computer hardware, such as semiconductor chip size, manufacturing costs, and reliability, along with advances in their communications capabilities, are essential for building ubiquitous computing systems. Fortunately, Moore's Law has already helped us to make inroads along many of these dimensions, giving us match-box-sized computers, fast processors, and wide-area networking capabilities[4].

What Moore's Law has not been helpful for, however, is the design of ubiquitous computing software applications, consisting of their user interfaces, supporting tool sets, and middleware [55].[5] To make ubiquitous computing systems useful and natural for people, these higher level designs must resonate with people's subjective preferences in the ways they like to work, communicate, and live. Yet, isolating these preferences and identifying how to design systems to fit them has remained challenging, largely due to a multitude of as yet unanswered questions about human psychology and physiology.

The burgeoning field of Human-Computer Interaction (HCI) seeks to make this human-centered design tractable by taking a scientific approach to discover what

reactive environment, or *aware space*

[4] Those unfamiliar with Moore's Law should refer to http://www.webopedia.com/TERM/M/Moores_Law.html

[5] *Middleware* is used to describe software that mediates between two or more disparate software components.

qualities make good designs for particular types of people. HCI accomplishes this through an iterative design discipline that involves first studying how people initially perform a task, without any additional technology, then, formulating an hypothesis of how technology could assist the task by realizing a prototype system (or mockup). Finally, users are asked to evaluate the system and recount their experiences of using the prototype. Users' feedback is used to improve the design, which is then again tested and revised. This process of iterative *user testing* not only serves to determine and maximize the *usability* of the system being designed, but also to form more generally applicable design principles.

The primary research focus of the Ki/o Kiosk Platform is to determine how to design effective and useful IEs for informal public spaces. This includes studying *how* users prefer interacting with kiosks, *what* users want to be able to do, and *whether* IEs for informal public spaces are fundamentally useful for people. These questions all fall within the research field of HCI, and thus the Ki/o Project takes an HCI, or *user-centered* design approach. Other design considerations, such as speed or accuracy of underlying algorithms, remain secondary concerns to the project.

1.4 Why Kiosks? Motivations for Kiosks as IEs

Public spaces play an important role in the social and professional interactions of people in the workspace. As will be described in this section, today's knowledge workers are required to work closely with one another, while facing environmental and social pressures in crowded, disjointed, or physically fragmented workspaces. Intelligent Environments that occupy public spaces, such as Ki/o Kiosks, seek to alleviate some of these pressures while promoting a sense of community.

1.4.1 Understanding the Knowledge Worker

The Need to Collaborate

In 1959, economist Peter Drucker predicted that the *knowledge worker*, a new class of skilled worker whose job is exclusively to create, manipulate, and disseminate information, would become a dominant figure in the work force of the United States by the end of the century [15]. The rise of the knowledge worker, he predicted, would spur radical social transformations by creating an upheaval of the existing skillsets, levels of education, and domains of specialization required for specific jobs. Knowledge workers are evaluated based upon how much they know, and how effectively they can learn new skills and concepts, rather than on exclusively what they produce. Since knowledge workers must be extremely specialized in order to be effective on most tasks, they must frequently rely upon collaborations with other knowledge workers. Drucker speculated, thus, that this rise of knowledge worker collaborations would promote new types of teams . . . teams that were small, self-organized, dynamic, and short-termed, and which crossed traditional organizational boundaries.

Peter Drucker's predictions have largely become true; knowledge workers currently comprise over one-third of the work force of the United States, and theirs is the fastest growing type of occupation. These workers currently command among the highest average salaries in the business industry, and require more skill and specialization than any other field [15]. Hoping to maximize their returns on their investments in these "gold-collar workers," companies have sought to understand how knowledge workers function, and thereby determine how best to increase their productivity. This has led to workplace studies examining sociological and psychological aspects of the knowledge worker's physical workplace, including their office design and layout, and studies of how knowledge workers communicate and relate to one another.

Among the findings of these studies are results that indicate the frequency and importance of small informal meetings. Related findings have indicated that knowl-

edge workers spend more time away from their desks, interacting in public spaces. These findings have served as the main impetus for exploring how augmenting public areas and informal meeting spaces with ubiquitous computing tools could ease and improve knowledge workers' day-to-day activities.

The Workplace As Meeting Nexus

One of the observed trends is that knowledge workers are increasingly deviating from standard 9-5 work days at the office, and are instead working remotely from home, late at night, or on weekends. The greatest contribution to this shift is currently believed to come from social and environmental pressures induced by shared office and cubicle workplace arrangements during the regular work-week, which deprive workers of privacy and control of their environments, as well as to avoid distractions. The arrival of remote collaboration and telecommuting tools has permitted a tentative means of escape, by allowing knowledge workers to work wherever and whenever they please.

The primary drawback to working independently and remotely is the increased difficulty for workers to meet face to face to work together, despite the increasing need for workers to collaborate. The result is that many knowledge workers have begun working in two phases: remotely, at odd hours when they require solitude, and at the office, when they need to connect with others. Thus, the role of the workplace has become a sort of meeting nexus, where knowledge workers come with the purpose of interacting with others.

The Creative Office

The rapid rate of technological progress over the past two decades has not only inspired a sudden, urgent need for knowledge workers, but has also placed extreme demands on companies to be flexible and accommodate for change. As Jeremy Myerson and Philip Ross describe in their book, *The Creative Office*, companies have had to drastically reduce the time-to-market for new ideas and products in order to

stay competitive. This has caused large companies, in particular, to have to restructure by foregoing the monolithic, controlling, factory-office style office management schemes of the mid-twentieth century for fast, flexible structures that can produce ideas quickly in order to drive ahead of the competition. Realizing that the knowledge workers' ability to think creatively and communicate with others is becoming a crucial determining factor towards a company's ability to be innovative, agile and successful, management in these companies has started to dissolve both physical and organizational barriers between employees. Physically, office space is being laid out to favor open, mobile layouts that encourage workers to move about during the day, instead of traditional, static, enclosed, personal offices or cubicles [33]. Organizationally as well, these companies have begun dissolving both vertical distinctions between levels of the corporate hierarchy, as well as horizontal divisions across departments, with the goal of eliminating any barriers limiting intra-office collaboration. Instead of formal organizational groups, these companies are fostering self-organized, informal collaborative groups. These collaborative groups tend to be small, transient, and dynamic; groups usually assemble exclusively for the purpose of solving or discussing a single, specific, design problem or topic, and dissolve after the problem has been solved.

These informal collaborative groups are usually formed spontaneously and randomly, often as the result of workers discovering common interests or another worker's expertise. Although there is no definitive way that these shared interests are discovered, they often have been attributed to random, chance encounters in informal spaces.

Random Encounters and Informal Meetings

A number of recently conducted workplace studies have provided evidence to support the theory that random, chance encounters in public spaces lead to informal collaborative meetings. These studies suggest that casual, social exchanges of nascent ideas in informal settings often trigger collaborative idea cascades. Scheduled large-group meetings that follow fixed agendas, in contrast, have been viewed

as milestones where information and developments can be formally disseminated to one's peers, but where few original ideas or feedback is generated.

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 enhance 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 for the workplace. Figures published by the study indicate that, on average, less than half of the participants' work-week was spent sitting at their desks. Instead, the remainder of the time was spent at meetings away from their desks, holding impromptu meetings "in secondary spaces, such as hallways, enclaves, and at water coolers" [51].

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 their design meetings, versus how they would like technology to 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 are not anymore conducted by meeting in a room, but by providing an environment and a situation where encounters happen." [53].

Standing During Informal Meetings

Other statistics collected by the Steelcase study revealed an increasing preference for conducting short meetings with members standing in informal group rather than sitting fixedly around a conference table. Participants reported that 64 percent

preferred standing in impromptu meetings rather than an alternative posture, such as sitting at a conference table, or leaning.

Since these findings suggest that informal impromptu meetings tend to be unplanned, and usually occur while standing in random public spaces, participants are rarely prepared with writing or recording tools to record notes. Therefore, people must try to remember and later recall the conversations they had. This is very difficult, particularly for more involved meetings, when multiple or technical ideas are discussed, or when a series of impromptu meetings occur successively. Personal experience has shown that all but the most salient topics frequently get forgotten before they have a chance to be written down. Ki/o was designed primarily to capture these spontaneous meetings, enabling one to log everything that is discussed to make it convenient for later access and retrieval. This application, the Opportunistic Meeting Capture system, will be discussed later.

1.4.2 Building Community Across Physically Disjointed Workspaces

Another motivation for designing IEs for public informal spaces is to try to bridge physically disjointed workspaces via *awareness services*. These awareness services try to build a sense of community by conveying local contextual information to remote sites through facilities such as live audiovisual links.

The need for awareness services has increased phenomenally over the past decade. Workspaces fitting increasing numbers of knowledge workers within their limited real estate, accompanied by growing concerns over the social effects of packing workers together in tight "cubicle farms" have forced fragmentation and expansion to satellite offices. A number of large corporations have also expanded to offices in disparate time zones around the world to keep the company productive 24 hours each day. This has increased the need for tying awareness services across physical space and time to encourage workers to remain socially in contact with their peers in remote locations.

One primary motivation for the Ki/o Kiosk Platform is to provide passive virtual

presence and awareness facilities by providing audiovisual views into remote public spaces. Ki/o also supports a community digital bulletin board application, which can be used to uniformly disseminate and coordinate information display across all sites. Telecommuting workers could also take advantage of the Ki/o facilities in order to gain a feeling of presence.

1.4.3 Kiosks for IE Research

Research in designing IEs for primary spaces such as meeting rooms, offices and classrooms has encountered a number of significant challenges. These challenges are rooted in the general-purpose nature of these spaces, combined with their flexibility to accommodate a large group of people using the space simultaneously.

Perception

One such challenge has surrounded the problem of perception, or accurately sensing and interpreting the state of users and the physical world. Perception is a crucial component of IEs and has the greatest potential to improve the overall user experience by enabling the systems to observe its users, and to deduce user intentions and desires automatically. This frees users from potentially having to explicitly input this information when it is needed. Perceptual data can be used to trigger automatic, reactive behaviors, or tailor and expedite interactions with the UI, such as by predicting what the user might do next. This can be compared to granting IEs the ability to sense and "do what's right,", as well as to "read" their users' body language.

Research in methods underlying perceptual systems falls within the very active domains of machine vision, sound processing, and sensory fusion. Unfortunately, even the most current techniques in these domains today run into difficulties with general purpose spaces such as meeting rooms and classrooms. These spaces accommodate groups of dramatically varying sizes, and within them, people can move about freely, and assume various postures while doing any number of various ac-

tivities. As a result, the task of accurately locating people within the space, and disambiguating their mutual auditory and visual signals for the purpose of determining contextual clues has remained inherently challenging.

Fortunately, kiosks feature constraints that help to reduce some of these perceptual challenges. These constraints arise from kiosks' small, fixed *interaction volume*, which consists of the physical region in which users must stand in order to interact with it. This is typically a function of the size of the kiosk's display, and mode of user interaction; kiosks typically require users to stand in proximity to interact with them, and have relatively small displays. Therefore, cameras and other sensors can be focused and optimized for users standing within this volume. A related feature is the maximum number of simultaneous users of a kiosk; the interaction volume limits the practical number of simultaneous users to two or three, thereby simplifying disambiguation.

An IE's perception systems consists of both hardware and software components. The design of the Ki/o sensing hardware, such as cameras, motion detectors, pressure and distance sensors, is described in Chapter 3. Data from these sensors, in turn, is collected and aggregated to form higher-level conclusions about the world. A specific software architecture design known as a blackboard architecture was chosen for this purpose, and is described under section 4.5.

Interaction Modeling

Interaction modeling is a process by which user interface systems learn predictive models of its users and their tasks to facilitate future user interactions. This can reduce the burden on users by saving them from having to repeatedly specify everything exactly as they want it done, by allowing systems to learn about its users from prior experiences interacting with them.

Interaction modeling on kiosks is absolutely critical, because it can potentially significantly reduce the *startup time*[6] to use the kiosk, and as a result, allow inter-

[6] *Startup time* is the amount of elapsed interaction time with a user before he or she is able to perform his or her desired task. Large startup times plague most traditional user interfaces today.

actions with the kiosk to be made vastly more quick and efficient. A user model can automatically supply information to an application that the user would have to otherwise explicitly provide with each interaction.

Modeling interactions with kiosks could potentially be simpler than in other types of spaces because most interactions with the kiosk will be brief and done to accomplish a single or small number of tasks. As described in the last section, restrictions on the practical number of people that can interact a kiosk will almost certainly simplify interaction modeling as well. As described in section 5.4.1, determining how to most effectively perform interaction modeling is one of the primary research questions left for further investigation.

Chapter 2

Background

The Ki/o project serves as an extension of previous IE research of the *AIRE* group at the MIT Artificial Intelligence Laboratory. This chapter provides the historical context of the previous work done by *AIRE*, and serves to indicate the Ki/o project's position within the context of MIT's broad ubiquitous computing project, Project Oxygen. It also presents a summary of related ubiquitous computing research done elsewhere that have been the most influential to the project.

2.1 Ubiquitous Computing at MIT: Project Oxygen

Project Oxygen is a united effort between the MIT AI Laboratory and the Laboratory for Computer Science (LCS) to make computers systems human-centered, omnipresent, and invisible [39]. Its name was devised by Michael Dertouzos, director of the LCS from 1974 until 2001, who aspired to re-unite technology with humanism and wished computers could become transparently available and accessible to anyone, anywhere, thereby seeming "as natural a part of our environment as the air we breathe" [26].

2.1.1 AIRE: Agent-based, Intelligent, Responsive Environments

Within Project Oxygen is a wide array of subprojects that span various disciplines in computer science. The *AIRE* research group focuses on technologies related to building Intelligent Environments. Using its distributed agent architecture, *Metaglue* [9], AIRE has built software architectures for intelligent environments in the forms of offices and conference rooms (e21), handheld computers (h21), and now, kiosks (Ki/o).

The Intelligent Room

The primary prototype test bed for the e21 architecture is embodied in a conference room known as *The Intelligent Room* [5]. A large number of projects have come out of work in the Intelligent Room, including Metaglue itself, the Metaglue resource manager, RASCAL [18], and a framework for specifying layered, reactive behaviors, ReBA [25][24].

h21: Next-generation Handhelds

As described in Chapter 1, today's workplace is becoming increasingly decentralized and distributed, and the need for the knowledge worker to continually be on-the-move is increasing. To enable the knowledge worker to access this information anywhere in the workplace, Project Oxygen created the h21 handheld Intelligent Environment. The h21 is the result of a collaborative effort with AIRE, the Networks and Mobile Systems (NMS) group in the LCS, and Compaq Corporation's Cambridge Research Laboratories (CRL). Like the other AIRE spaces, these h21s run a distribution of Metaglue, but on top of a minimal variant of the Linux operating system. Physically, these h21s consist of a Compaq iPaq handheld computer with extra components, including wireless Ethernet, memory, and occasionally a small camera and accelerometer [52]. In addition, the NMS group has developed an indoor, GPS-like location system for the h21 known as the *Cricket Location System*. With the Cricket, any h21 may constantly be aware of its own precise coordinates

and orientation within a building [42]. As will be described in Chapter 4, these Crickets can be used for location-based awareness and information services.

Ki/o: Kiosks for Informal Spaces

Ki/o kiosks, thus, constitute the third type of AIRE space, falling between e21 and h21 in size and complexity. As will be discussed in the remaining parts of this paper, the form factor and intended use scenarios of Ki/o kiosks produce a new set of design requirements unique from those of either e21 and h21.

With the introduction of both the h21 and Ki/o platforms, the need for Metaglue agents to be able to easily share information across IE boundaries has suddenly become extremely critical. This requires Metaglue to transition from a simple, flat local namespace for to a global, hierarchical namespace, and the need to create a means by which access control can be exercised between IEs. This cross-IE version of Metaglue is dubbed *Hyperglue* and is currently under development.

2.1.2 Perceptual User Interfaces

AIRE's partner group in Project Oxygen, the Vision Interfaces Project (VIP), has focused on designing perceptually driven user interfaces and developing robust vision algorithms for AIRE spaces. VIP has contributed person-tracking, face-detection, and head-pose tracking [31] systems to AIRE spaces, enabling experimentation with various types of perceptually-driven non-command user interaction [34].

2.2 Related Work: Kiosks

2.2.1 DIGITAL Smart Kiosks

One of the first inspirations for the Ki/o kiosk project was Digital Equipment Corporation's Cambridge Research Laboratories' Smart Kiosks project, which deployed a number of prototype kiosks around Boston. These vision-enabled kiosks featured

a series of unique animated disembodied avatars that would watch and summon users passing by. Large touchscreen CRT displays were used as the primary interaction method, while vision and synthesized speech were used for backup input and output channels. The tight integration among the various vision, avatar, GUI and application execution components made responsive, real-time interaction possible, and the researchers were able to gather thousands of hours of user interaction data from the experiment [8]. Since this project served as one of the first examples of an interactive vision-based perceptual information systems running in real-world non-laboratory settings, it served as an early social litmus test for camera-based vision systems in the future. Their results indicated that user response was positive, and that providing visual feedback of the vision algorithms in action alleviated most users' shyness or privacy concerns.

2.2.2 The IBM BlueBoard

Researchers responsible for the *Bluespace* ubiquitous computing office at IBM Research have developed interactive touchscreen public displays of various sizes known as *BlueBoards* for the purpose of group information sharing in public spaces. Users approaching a BlueBoard first authenticate with it using their standard corporate RFid tags, which then causes the BlueBoard to load up each individual's personal calendar, messaging and personal file browser. Large plasma-display BlueBoards are placed in prominent public spaces for simultaneous multi-person interaction, while smaller ones have been designed for outside personal offices to display office awareness information (such as person availability), and allow visitors to leave notes when the occupant is away [46].

The BlueBoard and Ki/o projects are similar in many respects, differing mainly with respect to particular research emphasis. As will be described later, the Ki/o project research emphasis leans toward perception-driven, non-command, agent-based user interaction, whereas BlueBoard seems to emphasize high-visibility, overt, direct-manipulation style interfaces. Furthermore, one of the ultimate primary ap-

plications of the Ki/o lies the capturing of informal meetings, whereas BlueBoard seems to serve more as a digital portal to a user's personal data.

As will be described in the next section, a third project, known as *Aware Community Portals* was a project to develop kiosk-like interactive wall projections from the MIT Media Lab for the purpose of community building and awareness. Since this project shared many of the goals of the Ki/o platform, it, too was important in defining the scope of the Ki/o project.

2.3 Related Work: Media Spaces and Awareness

2.3.1 Xerox EuroPARC: Portholes

Inspiration for the O2Flow Architecture, described in section 4.3 came from a number of early experiments with audio-visual links to tie disjointed physical spaces at Xerox PARC in the 1980s. The first such application which used media spaces for the purpose of *awareness* was the *Portholes* project in 1992 by Paul Dourish and Sara Bly at Xerox EuroPARC, who discovered that when cameras and displays were not actively being used for a videoconference, users wished to see displays of public spaces, such as near coffee pots and lounges. Dourish and Bly believed this desire was motivated by people wishing to find out what activities were going on in these spaces, such as to see who was there, who was talking with whom, who was available for interaction. This information, they believed, encouraged spontaneous interaction and contributed to the social health and sense of community in the workplace [14]. Unlike video conferencing systems and prior media space experiments, their system was intended to be viewed passively, in the background of user's tasks.

2.3.2 Media Lab: Aware Community Portals

Subsequent to the development of Portholes, a large number of projects incorporating awareness into groupware and media space applications have been developed.

Nitin Sawhney at the MIT Media Laboratory summarizes nearly a decade of research devoted to collaborative applications involving video and audio links in his paper *Situated Awareness Spaces*, including describing a project he participated in designing, called *Aware Community Portals* [47]. His impressive prototype integrates a large number of elements many of these earlier designs to provide both spatial and temporal awareness, large-group presence indication, and group interest awareness and display.

2.4 Informal Meeting Capture

The field of Computer Supported Collaborative Work (CSCW) has seen a wide variety of research projects dealing with both synchronous and asynchronous, colocated and telepresent prototype meeting capture and facilitation tools. Many of these focus on the whiteboard as the center of collaboration, such as Dynawall [19], Tivoli [40], We-Met [60], M-Pad [45], and DUMMBO [6]. Only one of these, however, has been targeted to augmenting unplanned, random informal interactions as might occur as the result of random encounters in secondary spaces. For their system, DUMMBO, Jason Brotherton, Gregory Abowd and Khai Truong from the Georgia Institute of Technology outlined how the design requirements for unplanned informal meetings differ from traditional meetings; for informal meetings, startup time and cost for using the system must be absolutely minimal, and, secondly, the system must support the inherent lack of structure of these types of meetings. This lack of structure makes later retrieval and organization inherently more computationally difficult for informal meetings, yet all the more important [6].

2.5 Mobile Devices

2.5.1 Xerox PARC: ParcTab

To get ideas about how mobile devices, such as Oxygen h21s or notebooks or tablet PCs might be able to seamlessly interface with Ki/os and other Intelligent Environments, a number of projects incorporating devices of various sizes and form factors were examined. *ParcTab* was an early ubiquitous computing infrastructure at Xerox PARC (designed in part by Mark Weiser) that consisted of devices in 3 forms, the "inch", "foot" and "yard" form factors, Even at the "inch" scale, the petite ParcTab mobile unit was integrated with the rest of the system, communicating over infrared. The "yard" unit was the Xerox Liveboard, which supported simultaneous, multi-person operation [57].

2.5.2 Carnegie-Mellon University: Pebbles

The *Pebbles* project, started in 2001 by Brad Myers at Carnegie-Mellon University, has sought to develop software that enables existing standard PDAs running PalmOS or WinCE to be used to control and augment traditional user interfaces on PCs. Examples of novel applications that have already been developed with the Pebbles architecture include a remote-commander application which provides convenient access to application macros, as well as convenient zoom, pan, and presentation control widgets, or remote scribble and clipboard manipulation functionality all via the PDA display [32].

2.5.3 GMD: iLAND and Roomware

Another demonstration of tight integration among heterogenous mobile components was done by the iLAND project, mentioned in section 1.4.1, from the German National Research Center for Science and Technology (GMD). Through their *BEACH* software, note contributions from anyone using any of their *RoomWare* components could be shown, moved, duplicated and manipulated in a direct-manipulation man-

ner through the use of dragging and throwing gestures. Their simple and elegant solution for cross-device information sharing and collaboration seems appropriate for both informal and full meetings alike [53].

Chapter 3

Physical Design

This chapter describes the physical aspects of the Ki/o kiosk IE. It presents a tour through the design process that was used to arrive at its current design, and provides the rationale used to evaluate alternative designs.

3.1 Identifying the Physical Form

The task of designing the physical form of IEs for transitional spaces requires consideration of creative, aesthetic and sociological factors as well as practical and logistical ones, such as physical and budgetary constraints. Many of these characteristics are specific to the site where it is to be installed, such as the size and shape of the space allocated to the installation, and what affordances the space provides for embedding and mounting equipment into floors, walls, and ceilings.

Due to the number of variables dependent on the installation site, no one design for Ki/o would fit all environments. Although this means that Ki/o IEs have to ultimately be designed on a site-by-site basis, this can be made simpler by considering multiple typical installation scenarios and determining what elements these have in common.

For insight on the different types of spaces and installation scenarios, the design process began with a survey of public spaces around the MIT campus that have been augmented with various types of digital information displays.

Figure 3-1: The List Visual Arts Center Media Test Wall

3.1.1 Survey of Existing Information Displays

The survey yielded a number of interesting displays which ranged widely in size, integration, and interactivity. The List Visual Arts Center (LVAC) Media Test Wall was the largest display that was surveyed, and consisted of an approximately six-foot square rear-projection screen deeply installed in the wall, coupled with a sound isolation dome speaker hanging from the ceiling (see Figure 3-1).

This display, which presently remains active, occupies an entire section of wall connecting the entrance and elevator lobby of the building to the main hallway. The open space provided by the lobby as well as its proximity to the nearby computer cluster and classrooms yields high visitor traffic. This display presents a rotating exhibition of the LVAC's video art collection, and is therefore non-interactive. The sound dome focuses the audio to observers standing under the parabola, and attempts to minimize noise pollution to the surrounding space. The result is a successful, eye-catching installation that captures the attention of passers-by, while respecting its surroundings.

A dramatic contrast to the LVAC display were kiosks recently installed in the

Figure 3-2: KIS Kiosks in the Aeronautics and Astronautics Department

newly renovated Aeronautics and Astronautics Building. These kiosks, designed by Kiosk Information Systems Inc (KIS) are much more conservative in their design, in that they do not attempt to depart the traditional notion of an information kiosk (see Figure 3-2). Since these kiosks are self-standing units, they are are simple to install, replace, and upgrade, but unfortunately appear less integrated with the environment. Equipped with a small screen, keyboard and mouse, the designers of the kiosk intended this to be used by a single person at a time, standing directly in front of the kiosk where the keyboard and mouse may be reached. Unlike the LVAC display, these kiosks were placed in smaller corridors and stairwells, and therefore received vastly less visibility and traffic.

A third display, at the Microfluids Technology Laboratory, shares many of the most desirable properties of both of the former two displays. As visible in Figure 3-3, its large plasma display device is eye-catching and permits comfortable viewing from a distance, while a trackball mouse welcomes users to approach the display and to interact with it. However, unlike the KIS kiosk, the size and horizontal aspect ratio of the display area suggests that a small group of two or three people

Figure 3-3: Microfluids Laboratory Hallway Display

could simultaneously gather around the display, such as for a collaborative discussion. Furthermore, by being installed in a larger glass case embedded in the hallway wall, this display manages to appear well integrated with the environment, while maintaining the flexibility of having its components later replaceable and upgradable.

3.1.2 Physical Design Criteria

This survey of existing displays facilitated identifying qualities desirable for the Ki/o kiosks, as well as determining the most relevant dimensions upon which designs could vary on a per-site basis. The most important qualities and dimensions are discussed below.

Display Area

The size and shape of the display has the greatest impact on how many users can simultaneously view it, as well as how close users have to be. While larger displays have many obvious advantages they are also exponentially-by- size more expensive to acquire, install, and maintain. The current best large-screen display technologies are projection and plasma displays, the former of which suffers from extremely high maintenance costs due to short bulb lifespan, while the latter suffers from low visual resolution and extremely high costs. Therefore, until a single viable display with all the desirable qualities (high resolution, low cost and low maintenance) is available, the type of the display must be chosen on a per-site basis.

Form and Installation Technique

How a display is presented in its environment shapes new users' impressions of a system and affects their *a priori* expectations for the system's capabilities and pleasantness of use. These expectations are usually derived from associations of prior experiences with other, more familiar systems. In order for users to draw the appropriate associations and expectations, a physical form must be chosen which

suggests a connection to an appropriate familiar system, while simultaneously appealing to users' aesthetic sensibilities.

This is a challenging task because the installation technique will be invariably dependent on the affordances of each installation site. On the one hand, the display should appear seamlessly integrated with the environment in which it is installed. This may demand that display designers work with building architects to make plans for the display prior to building construction or during renovation. However, most designs will not have this luxury, and extensive building modification will be logistically impossible. An example of an alternative approach is the installation technique for the KIS kiosks described earlier. In this case, installation involved little more than merely strategically placing the self-standing structure in an open space. While a tighter integration has aesthetic benefits, the KIS kiosk lends itself to practical benefits such as ease of installation, equipment access, maintenance, and upgrading. As described earlier, the Microfluids display installation provides a balance by permanently embedding a glass case in the wall, which provides the flexibility to allow its contents to be changed easily while providing the aesthetic appearance of seamlessness.

Although ultimately designs must be considered on a site-by-site basis, installations may be evaluated on the basis of their practicality, lack of requirements for difficult building modifications, and degree of aesthetic integration with its surroundings.

Mode of User Interaction

Another important consideration is the question of what hardware devices the kiosk will use to perceive users' actions. This choice plays a key role in the user experience, by determining how easily users can communicate their intentions and desires to the system.

In the three displays surveyed, only the KIS and Microfluids display were interactive, and both of these used traditional desktop input devices, specifically keyboards and trackball mice, to provide input. Although this may be sufficient for the

KIS kiosk due to its similarity in form factor to a PC, these input devices are usually extremely clumsy, non-ergonomic, and limited when used away from the desktop. Among the problems are an inability for multiple people to interact with the display simultaneously. Various alternative input devices will be discussed in greater detail later in section 3.2.

Criteria Specific to IEs

The design criteria heretofore discussed are relevant to designing any interactive digital public information display. However, since IEs also have facilities for perception and action, they usually have a number of additional requirements that dictate further physical design constraints.

These requirements specific to IEs comprise those needed by perception and sensing hardware, which may consist of monocular and stereo cameras, motion detectors, and distance and pressure sensors, as well as the ability to move and change its physical embodiment or surroundings, such as via motors and servos. For example, Ki/o IEs, using a set of cameras or motion detectors, could determine how to best move its displays when a user approaches, such as placing them within reach of a user in a wheelchair.

The first generation of Ki/o IEs that have been designed to accommodate flexible experimentation with a wide variety of hardware devices, so that a variety of sensors, cameras, and actuators could be tried and evaluated. Thus, modularity has been favored over aesthetic value or tight physical compactness. The various physical prototypes will be described and illustrated in section 3.3.

3.2 User Input Devices

Traditional input devices such as the keyboard and mouse have evolved for use on desk surfaces and are designed and optimized for the ergonomics of desk surfaces. While in some situations it may be unavoidable to use a keyboard and mouse at a kiosk, alternative input devices can provide a far more efficient physical inter-

face for the transient types of interactions that are typical of kiosks. The following discussion addresses some of these alternative input devices.

3.2.1 Tangible Input Devices

Tangible input devices, such as those popularized by Hiroshi Ishii and Brygg Ullmer at the MIT Media Lab seek to simplify human-computer interaction by extending user interfaces and abstract digital entities into the physical world [23]. Their research surrounds the vision that if digital information can be embodied in familiar physical forms, such as blocks, tiles, containers, or post-it notes, they can be directly manipulated easily by practically anyone.

By tightly coupling physical and digital entities, Ishii's prototypes have successfully demonstrated significant reduction in the *gulfs of execution and evaluation* [1] which plagues most UIs in existence today. Furthermore, they reduce the learning curve required to manipulate user interfaces, by literally providing a physical metaphor for abstract actions.

However, creating effective tangible input devices requires that the designer have access to prior knowledge about the types of data to be manipulated. If there are multiple such types of data, a physical surrogate must be designed for each.

Realizing that Ki/o IEs may embody a wide assortment of types of data ranging from awareness-related information to captured meetings and notes, no single model could be isolated that could be realized as a physical surrogate or set of surrogates. Therefore, designing a tangible user interface was deemed too difficult, at least for the first design. Instead, other, more generic direct-manipulation interface devices such as touchscreen displays were considered.

[1] a term introduced by Donald Norman with respect to UIs to refer to the (1) cognitive gap between wanting to do something and being able to actually do it, and (2) the difficulty in determining whether or not the desired output was actually achieved.

3.2.2 Touchscreen Displays

Touchscreen displays enable users to interact with displays directly by touching and tapping their surfaces. Touchscreen technology has evolved over the past few years improving in sensitivity and pointing accuracy, moving from traditional resistive sensing to capacitive sensing techniques that can detect a hand even as it approaches the display surface. Several manufacturers, such as 3M have developed simple glass overlays that fit over the surface of standard CRT or LCD displays, transforming them into touch-sensitive displays.

From a user standpoint, touchscreens provide a simple and obvious mode of interaction that requires little learning. However, as discovered during the initial user evaluation period of the Ki/o prototype, if steps are not taken to redesign software that were originally optimized for mouse interaction, the user interface can seem unresponsive and frustrating. This is largely attributable to the difference in control precision provided by a touchscreen versus a mouse; mice can be accelerated or decelerated nonlinearly to provide rapid navigation or fine-grained control, whereas touchscreen displays provide a linear, one-to-one mapping between physical and virtual space. Another difference involves how taps map to mouse clicks; many users will not make a distinction between glass presses and taps.

3.2.3 Sensing and Perception

IEs such as Ki/o have perception subsystems made to provide the capability of sensing and perceiving their physical surroundings. Perception has the potential to greatly enhance the user experience by enabling the system to automatically deduce what the user wants by gathering and analyzing contextual clues, much in the way that humans read body language. Unfortunately, much of this field is still an area of open research, and is one principal areas of exploration in designing IEs such as Ki/o.

In order to perceive the world, IEs must rely on sensors of various types and forms. Data from low level sensors, such as motion detectors, pressure and distance

sensors can be aggregated to form higher-level conclusions about the world. A software architecture for doing this known as a blackboard architecture, will be described in section 4.5.

3.3 Design Prototypes

Four Ki/o prototypes of varying forms have been designed for different spaces around the AI Laboratory. These range in size and functional capability, from the complete Lobby prototype to the minimal embedded wall display.

3.3.1 Lobby Prototype

About the Space

The first Ki/o prototype was designed for the eighth-floor elevator lobby of the LCS/AI laboratory building. This space was chosen for its proximity to the AIRE lab space, and for the large number of people that regularly pass through the space. Approximately eighty faculty, researchers, and staff members' offices reside on the floor, comprising roughly ten research groups. Visitor traffic is particularly high when gatherings are held in the eighth floor Playroom for lab-wide research colloquia and meetings of the GBACM (Greater Boston branch of the ACM).

The lobby's proximity to the AIRE research group facilitates monitoring and regular maintenance of the IE. It also permits monitoring of community reaction to the system, and encourages communication of feedback back to the Ki/o designers who reside nearby.

The lobby features thick brick walls that have remained basically unmodified since the inception of the laboratory in 1969. Although these walls make it impossible to embed anything within them, a more recently installed drop ceiling provides convenient hidden space for obscuring equipment and wiring.

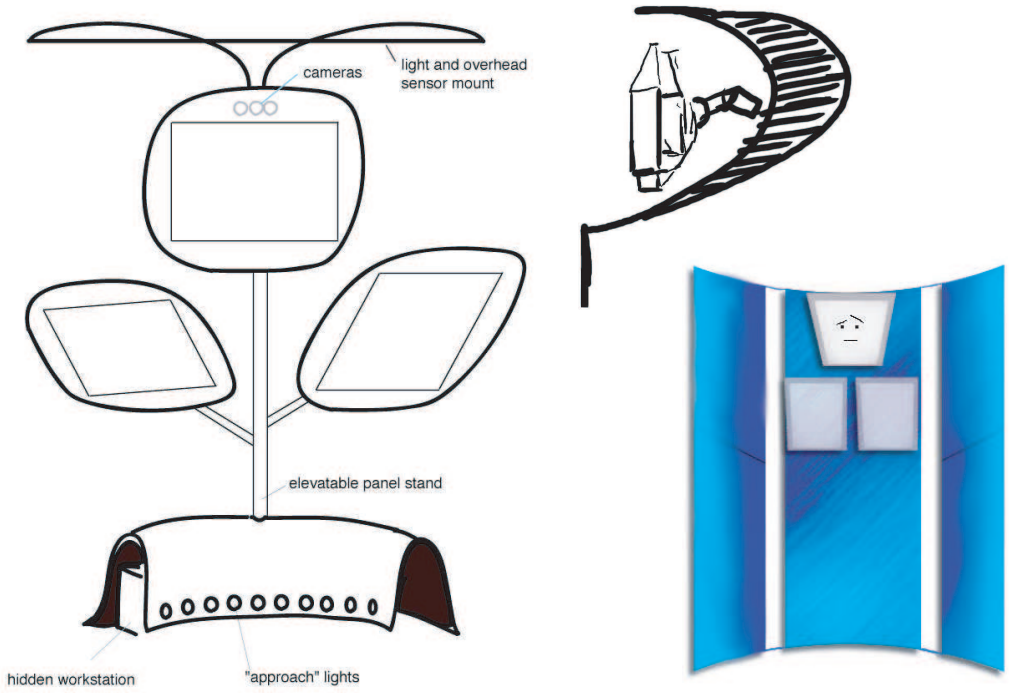


Figure 3-4: Early designs for Ki/o

Choosing a Form

Although the solid nature of the walls seemed to controvert the design goal of a seamless integration the surrounding environment. The problem of how such an integration could be achieved with these constraints became the primary objective.

Several floor-standing installations resembling a number of variations of the KIS kiosk concept were initially proposed (see Figure 3-4). Most of these were discarded on the basis of being too large and imposing, and for lacking integration with their surroundings. A more minimal and purely functional design which could better connect the kiosk to its environment was sought.

During this design process, construction of the new Stata Center, a building designed by Frank Gehry and Associates to replace the current building holding the LCS/AI lab, had just begun across the street. The freshly poured base concrete structure of the Stata Center revealed the building's dramatic inner angled pillars

and irregular shapes that constituted the building's unconventional, post-modern design style. Inspired by Gehry's design and hoping to convey a feeling for the design of the AI laboratory's new home, a new Ki/o design was proposed.

This new design (see Figure 3-5) uses long aluminum rods to mount Ki/o displays and equipment to the lobby wall. Two touchscreen 17" LCD displays are mounted on one pole each, which crisscross diagonally and meet the ceiling and wall at different angles. The hollow poles allow for the channeling of all display and device-related cabling up through the poles into the drop ceiling where it is connected to the remaining Ki/o support hardware concealed in the ceiling. The poles are intended to deliver the appearance of being somehow structurally connected to the building, and thus being architecturally part of the lobby, while minimizing modifications to the walls. A small monocular camera mounted above the LCD displays senses passers-by and approaching users, while a standard desktop microphone senses ambient sound. An isolation dome speaker hanging above the displays was chosen to deliver stereo sound to users standing underneath, while minimizing reflection outside the immediate area.

Display Requirements

While a wide variety of display options were under consideration, a number of factors influenced the choice of using a pair of standard desktop LCD displays for this first prototype lobby installation.

A lobby installation such as this required a display that could be noticeable and discernible from up to approximately 30 feet away, but which also provided enough resolution to support a readable display of text for users standing nearby and interacting with it. The display also had to be large enough to allow small groups of people to stand around it, view it, and interact with it simultaneously comfortably. A third important practical consideration dealt with the cost of each display; since displays have become the most expensive single component of computer systems today, the ability to build multiple Ki/o prototypes given a particular budget rests largely on the cost of the type of display chosen.

Figure 3-5: Ki/o Lobby Prototype

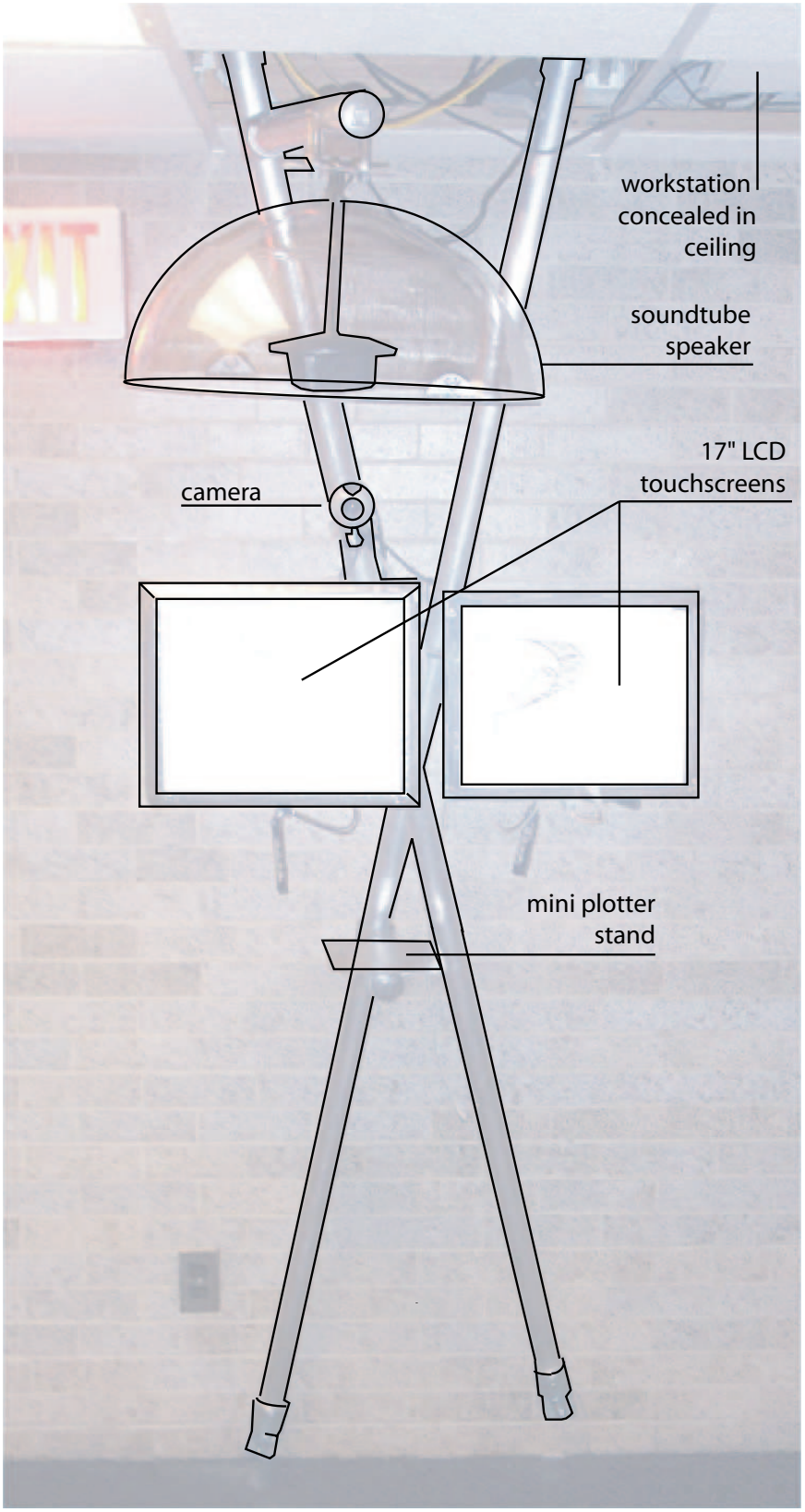


Figure 3-6: Ki/o Lobby Prototype Components

Plasma displays today come in sizes and form factors ideal for far-away visibility and supporting nearby multi-person interactions. However, low pixel density and their extremely prohibitive cost led to the adoption of LCD displays instead.

Desktop LCD displays support resolutions and pixel densities higher than plasma displays and comparable to that of traditional CRTs. Unlike projection-based displays, LCDs are also high-contrast (typically 500:1 contrast ratio). The main limiting factor with LCD displays is their size; current displays only achieve a maximum of 23 diagonal inches.

One technique to expand display surface area of LCDs is to create one large logical display out of multiple independent LCD displays. Unfortunately, since each display requires its own frame buffer, graphics accelerator, and video signal driving circuitry, this technique is limited by video card hardware scalability constraints. Fortunately, with the advent of increasingly powerful video cards, native multi-head support from a single card is becoming commonplace. Furthermore, since LCDs do not require analog signal feeds, video cards providing a direct digital video interface (DVI) can omit RAMDAC chips used to drive the displays. In the future, elimination of RAMDAC chips could provide enough real estate and cost savings to warrant supporting even more digital displays. Thus, tiling small, affordable desktop LCD displays to form larger composite display surfaces seems the current most cost effective, flexible option for the near future.

Audio Output

Many Ki/o applications are likely to need Ki/os to be able to produce audio output. However, since the lobby prototype is in a high-traffic gathering area, too much audio output could contribute to noise pollution which could be detrimental to interactions nearby. As a potential solution, a sound output device was sought that could focus sound exclusively to within the region of interaction, and limit external leakage.

A commercial product that seemed to fit these requirements called the Sound-

Tube[2] consisted of an upwards-facing speaker which projects sound into a large plastic dome. Although this dome is supposed to focus the sound directly for a person standing underneath, and thereby minimize leakage outside the region, in practice, the degree to which this was effective did not meet expectations. However, users frequently commented on the unusual appearance of the device, making comparisons to *The Cone of Silence* from the popular 1960's television series, "Get Smart". Due to its effectiveness in attracting the attention of passers-by, use of the SoundTube device has continued on the lobby prototype, both as a sound output device and as an aesthetic accent.

User Input

As just described, three main input devices were chosen for the Ki/o Lobby prototype. Touchscreens provide the main mode of user interaction, while a simple, VGA-resolution color USB QuickCam camera and microphone provide perceptual inputs. As will be described in section 4.4, the perceptual input from the camera is processed through a pipeline which extracts information about whether and how many users are nearby the kiosk. At the time of this writing, the microphone is only used to transmit ambient sounds as awareness information via O2Flow. However, plans for using the microphone for speech-driven user interfaces using the Galaxy natural-language dialog system are discussed in section 5.2.

3.3.2 Lounge Prototype

Although the Ki/o lobby prototype is the only deployed Ki/o system thus far, plans are underway for immediate construction of several other Ki/o kiosks. The first of these is for installation in an informal break area in the AIRE research lab space, visible in figure 3-7.

This Ki/o installation will differ from the lobby prototype in several significant ways. First, users will most likely be interacting with it in a group, sitting around

[2]see http://www.soundtube.com/

Figure 3-7: Lounge setting for next Ki/o prototype

the glass table. Therefore, interaction sessions with this kiosk are likely to span a longer time, and potentially involve more people simultaneously than the lobby prototype. At the same time, since this lounge is not within a *primary circulation* route of the building, it will likely receive less visitor traffic than the lobby. Finally, since the users will be sitting, interaction must be done at a distance instead of directly through a touchscreen tactile input.

To address these new demands, a projector and screen were chosen as the primary display medium and surface for the lounge. As visible in the figure, the large projection area should allow all users to be able to see and read text on the display effectively. For interaction, a small camera functioning as a laser pointer tracker was chosen because it enables the use of a laser pointer to simulate a mouse from a distance. Once this prototype is built and deployed, both choices will be evaluated for usability and effectiveness.

3.3.3 Hallway Prototype

The initial Lobby prototype design was the first example of how a Ki/o kiosk be installed in a transitional space, such as a hallway or elevator lobby. However, for many hallways, this design would be inappropriately large, and would not fit the design goal of being low-profile and unobtrusively integrated into the environment.

One of the next versions of Ki/o will attempt to be low profile and occupy as little public space as possible. The installation visible in Figure 3-8 illustrates a simple implementation which takes advantage of indentations in a wall for storing bulky desktop computer hardware, thereby intruding as little as possible on hallway space. This particular kiosk, equipped with a touchscreen, speakers, keyboard and mouse is used to allow the *MoreMagic* server administrators at the AI Laboratory to monitor the status of all servers in the building, using the NetSaint suite of diagnostic tools. The keyboard and mouse allow administrators to use the kiosk directly as a terminal console when problem arise.

An alternative design is visible in Figure 3-9. This display, which is actually a

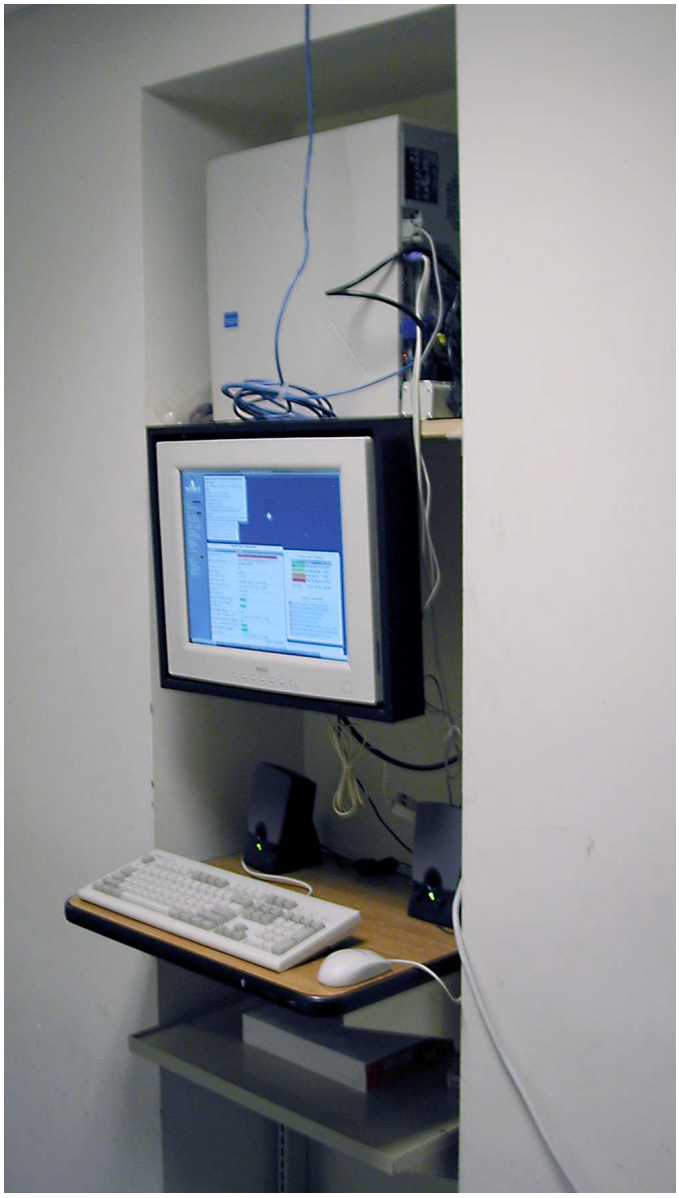


Figure 3-8: MoreMagic Network and Server Diagnostic Console Kiosk

Figure 3-9: Ki/o Wall installation

part of the Intelligent Room, is mounted directly onto a hollow wall which conceals cables running to a desktop computer in a neighboring machine room. By only exposing components that need to be visible, this is perhaps the most elegant design, but also has the most elaborate requirements due to the need for a custom wall and adjacent machine room. In the future, embedding necessary computer hardware within the display itself may eliminate the need for both the special wall and a nearby machine room. Tablet PCs equipped with wireless Ethernet could potentially be ideal for this purpose.

Chapter 4

Software Architecture

The demands of ubiquitous computing create many challenges for software design and development. As the research field matures, it has spawned new software design strategies and new programming models, repeatedly identifying and redefining design goals along the way. While many of these techniques have been tested on an experimental, individual basis, few of them have been integrated into larger systems and tested for actual use. The goal of the Ki/o project, therefore, is to consolidate and tailor existing methods across research domains into a software architecture that can be built, deployed in a real setting, and evaluated by users.

This chapter describes the the Ki/o software architecture, a prototype software toolkit for IEs installed in informal, public, or transitional spaces. The chapter begins by introducing the initial software design goals, and follows with sections devoted to detailed descriptions of the individual software components newly introduced by this project into the Metaglue system. The chapter concludes with a discussion of how these components interact with each other and with pre-existing Metaglue components, and provides a glimpse at several prototype end-user Ki/o applications built using this software architecture.

4.1 Architecture Design Goals

The purpose of the Ki/o software architecture is to serve as a toolkit to simplify development of end-user Ki/o applications. Thus, the software architecture aims to fulfill a role similar to that of contemporary user interface and application toolkits, such as Microsoft's MFC (Microsoft Foundation Classes). However, in addition to providing standard tools available in such a toolkit, such as GUI and communications capabilities, the Ki/o software architecture must support the needs of ubiquitous computing applications in intelligent environments. These capabilities may include speech recognition, natural language understanding, and visual perception.

To help identify the needs of these ubiquitous computing applications, this section presents the application design goals at a high level.

1. *Ubiquitous access to information and services*

   To achieve the ubiquitous computing objective of allowing computation to be available anywhere, regardless of the physical machine the user is interacting with, the Ki/o software architecture should permit all user data and IE services to be accessed transparently from anywhere. Transparent data access implies users of Ki/o kiosks do not have to worry about remembering which computer they left their data on, or how to transfer the data to their desktop or another IE. This is a problem that can be solved using a combination of a proper application/UI model, with network computing, which assumes that a high-speed network is available to connect all of the machines with which a user may interact, including mobile, handheld devices[1].

2. *Location and context-based information services*

   Ki/o UIs should be able to provide information relevant to the user's physical location and context. One example of such a *situation-based service* would

[1] A good example of ubiquitous data access can be seen in the *the Athena computing environment* at MIT. Athena workstations distributed throughout campus, which vary widely in architecture and specifications, present users with the same view of their data and available application software wherever they go.

be a Ki/o located near shared public printers indicating the current printer and print queue status when approached by a user who has a print job in the queue. The Ki/o could further permit manipulation, reordering, or removal of jobs from the queue, as well as possibly allow the user to contact others whose jobs are in the queue as well.

3. *Transparent flow of user state and task context between devices and spaces*

   The Ki/o UI should support *user state migration*, meaning that users current work state and task context should be able to migrate between spaces, devices, and IEs, following the user. While devices and spaces have such widely varying computational resources, display capabilities, and input devices, that the user work context may not be fully realizable in a particular device, they should still be restorable when the user has entered a space where such capabilities once again exist.

4. *Multi-modal Input and Multimedia Output*

   Ki/o UIs should support navigation and interaction via multiple modalities, such a via speech or gesture. To respond, they should likewise be capable of utilizing channels of communication that are most appropriate for the situation, such as text, video, graphics, or speech. Together, these capabilities increase *perceptual bandwidth*[2], or efficiency of information transfer between people and a user interface, by enabling users to express their desires more naturally, easily and succinctly, and, in turn, to receive information from the system in a clear, easily understandable fashion by using its most natural representation.

5. *Non-command Interaction and Reactive Behaviors*

   Non-command interfaces[3] treat users' time and attention to be its most important and scarce resource. These interfaces attempt to infer what the user

[2] also known as *interaction bandwidth*, see [44] for a more complete definition

[3] sometimes compared to *Do-What-I-Mean interfaces*, demonstrated by the Interlisp programming system[54]

wishes to accomplish, and actively tries to assist or automatically perform actions without explicitly being commanded to do so[4].

Reactive behaviors, which are actions triggered directly by perceptions, constitute the simplest form of non-command interaction. An example of such a reactive behavior would be a program that automatically turns on Ki/o displays in response to sensing that a user has approached. The theory of *subsumption architectures* supports the idea that any complex behaviors, including those responsible for human cognition, can be modeled by multitudes of primitive reactive behaviors interacting and suppressing one another[4].

The Ki/o should at least initially support reactive non-command behaviors, and later support more elaborate, task-specific non-command interaction.

6. *Direct-manipulation interaction*

   The philosophy behind *direct-manipulation interfaces*, as discussed in the last chapter, promotes putting users more directly in control of their systems, by reducing automation and abstraction surrounding data manipulation. While this seemingly contradicts the non-command based interaction philosophy, studies of direct manipulation interfaces underscore the importance of making systems fully visible, understandable, and easily manipulatable by users. The design of Ki/o applications should attempt to capture these qualities.

7. *Awareness Displays*

   As described in Chapter 1, one of the motivations for the Ki/o project was to attempt to build a better sense of community among members of an organization or workgroup. This goal is increasingly important for physically fragmented workspaces such as workgroups with remote or satellite offices. In these situations, it is highly unlikely for any members from the satellite office to casually encounter people from the main site in an informal setting,

[4] A widely familiar example of an early non-command interface is the Office Assistant available in recent versions of Microsoft Office, which used Bayesian inference techniques to determine what actions to take based upon what the user was doing.

and vice-versa, causing both locations to drift apart socially.

Awareness displays attempt to connect disjoint workspaces by capturing and transmitting information about local workspaces, and re-displaying them at the remote sites in a non-obtrusive way. One example of an early awareness display was the Portholes project[14] which, regularly took still, low-resolution photos of offices and public spaces in a laboratory and displayed them on every workgroup member's desktop. This permit any of the members to automatically tell whether someone was at their desk, and whether or not they were available for conversation.

To support awareness displays, the Ki/o software architecture should support a mechanism of capturing and transmitting awareness information to remote IEs, including other Ki/o kiosks. At the same time, the architecture must include facilities for allowing users to easily opt-out of sharing awareness information to provide privacy when desired.

4.2 Software Organization

4.2.1 Agent-based Programming: Metaglue

The Ki/o software design is an amalgamation of components each designed to serve one or more of the application design goals discussed earlier. These components are constructed and connected using an agent-based programming framework known as *Metaglue*, designed for originally for the Intelligent Room.

Unlike traditional software systems, the Metaglue agent architecture in the Intelligent Room has no central control logic or structure. Instead, each agent, an independent software entity with its own state and code, embodies a primitive, simple behavior. As Michael Coen, one of the original architects of Scatterbrain, a predecessor to Metaglue, describes, the theory is that when these simple agents are assembled and permitted to communicate with one another, more complex, aggregate behaviors can be achieved even without central control[10]. He further

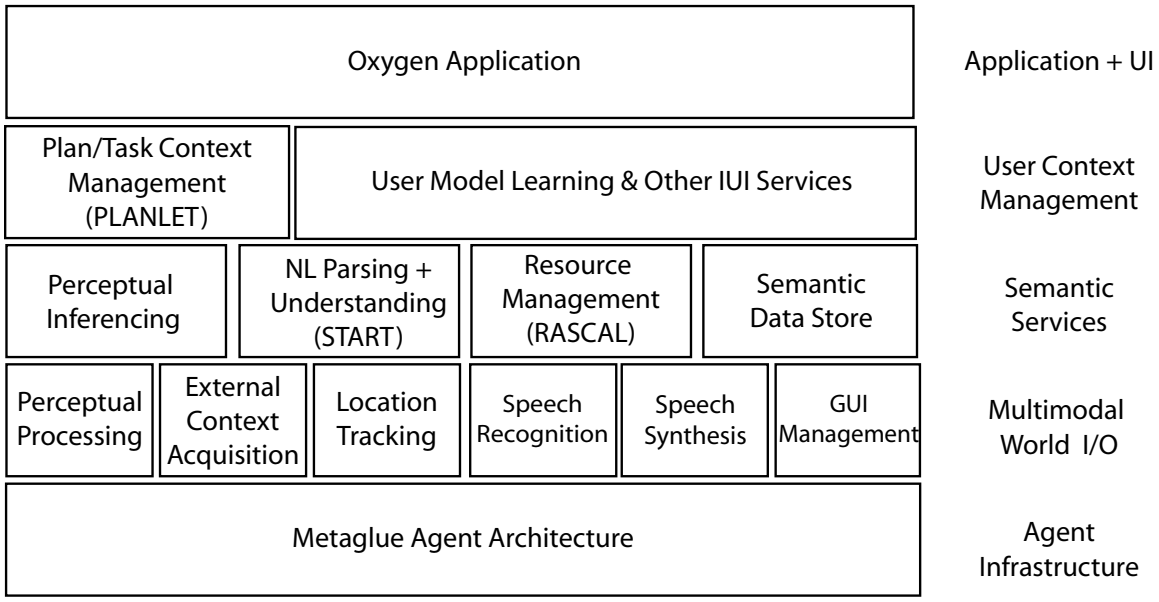


Figure 4-1: Ki/o Software Architecture Organization

argues that, in fact, this technique of distributed control is more fault-tolerant, by not relying on a single point of failure, and scalable.

With this agent programming approach, adding a new capability to the Metaglue framework consists simply of adding agents that embody those capabilities, and that fulfill the basic Metaglue communication capabilities to permit communication with other agents. Thus, the capabilities added to Metaglue for Ki/o, responsible for tasks such as perceptual processing, context aggregation, and media streaming, took the form of collections, or societies, of Metaglue agents. These individual agent societies are described in depth next.

4.2.2 Ki/o Services

Although Metaglue agents generally communicate with one another on an ad-hoc, peer-to-peer basis without any strict organizational structure, for illustrative purposes it is useful to categorize the various agent societies by the services they offer.

Figure 4-1 organizes the Ki/o agent architecture by levels of abstraction for the types of services offered. These layers are briefly described as follows:

1. *Metaglue*

   At the bottom is Metaglue, which provides all the facilities for low-level agent management and communication. Its primary responsibilities consist of controlling the birth of, and keeping track of, all live agents in a system. This information is then used to permit agent intercommunication, by providing applications with a layer of abstraction so they never have to explicitly know the location, identity, or status of each particular agent in order to communicate with them. Metaglue accepts queries for looking up agents based upon *capability* as well as explicitly by name, providing an additional layer of abstraction.

   A detailed account of the Metaglue system, its history, architecture, and design can be found in [41] and [58].

2. *Multi-modal World I/O*

   This layer provides other Metaglue agents interfaces to the physical world through various channels, including speech, video, and audio. The input services provide abstractions around keyboards and mice, sensors, video capture devices, low-level location sensing (Cricket) devices, and provides low-level signal processing services for these devices, such as speaker identification and speech recognition. For output, the GUI Manager controls GUI construction, and allows GUIs to be routed and reformatted for different sizes and types of displays. An alternative text output channel might be *text-to-speech* (TTS), computer-generated synthesized speech.

3. *Semantic Services*

   The next layer consists of higher-level *semantic services* which serve to aggregate and interpret knowledge in a variety of ways. The perceptual inferencing system receives raw data from visual trackers (cameras) and sensors, and drives towards possible interpretations of this raw data. The Ki/o software architecture uses a *blackboard architecture*[36][35] which will be described in

section 4.5 to provide this perceptual goal-directed inferencing capability. A *natural language understanding* (NLU) engine called START extracts semantic interpretations from raw English text and is also able to output semantic concepts as grammatically correct English sentences. A semantic store provides an organized, permanent storage service for semantic information and permits retrieval in a number of ways. The *resource manager* is responsible for fulfilling requests for abstract services and functionality with concrete agents that can provide that service, and resolves contention when resources are limited[18]. For example, an agent wishing to display data formatted in HTML may query RASCAL for an agent with such capability. The resource manager, in turn, could analyze the situation and respond with the most appropriate agent, possibly a standard web browser or perhaps an HTML page-to-speech reader.

4. *User Context Management*

   The User Context Management layer provides common services needed by oxygen (ubiquitous computing) applications and their intelligent UIs. Its responsibilities include transforming contextual information provided by the lower levels into models of the user and what the user's goals. These models can then be used directly by top-level applications to drive application state and to personalize the UI. The PLANLET project provides a portable way of explicitly representing user tasks. Using this representation will be one way to fulfill the goal of enabling the flow of task context between spaces.

5. *Application*

   The Application layer consists of the software pertaining to the actual task the user wishes to accomplish. Much like traditional applications, Ki/o applications provide core high-level logic and abstractly relying on lower-level software services. Several trial Ki/o applications, including a web information display and an interactive 3D floor plan are described in section 4.6.

4.3 O2Flow Awareness Network

4.3.1 Design goals

The O2Flow network was intended to simplify the acquisition, dissemination, and reception of various types of awareness information. Examples of such data may include low-level proximity and motion sensor data, still images from cameras, or live video.

This type of awareness information exhibits several characteristics that distinguish it from other data transmitted in Metaglue :

1. *Time-sensitive, non-essential delivery* - Awareness information is almost always extremely time-sensitive, but at the same time is not absolutely critical for delivery; therefore, if awareness information is lost, it should not be re-transmitted.

2. *One-to-many reception* - Since awareness is a type of notification mechanism, the messages will be intended for a one-to-many type of transmission.

3. *Stream-based payloads* - Many awareness sources, particularly live video or audio, are best modeled as continuous data streams rather than as individual messages.

While the Notification mechanism[58] already part of core Metaglue provides a means for message broadcast to all agents with a society, it is entirely discrete-packet message based. Using this transport would not suit stream-oriented data such as video, since the sheer number of messages required to sustain a continuous video stream would overload the mechanism. Furthermore, the notifier provides a guaranteed-delivery contract, which is not necessary for awareness information.

Thus, a more lightweight, stream-compatible dissemination architecture was sought, and the yielded the O2Flow architecture, described next.

4.3.2 Design Description

The simplest possible design was sought for achieving the goals above. Fortunately, the standard Java libraries provided capabilities that allowed made the design and implementation straightforward.

The network transport chosen to achieve time-sensitive, non-essential delivery is the standard unicast datagram protocol (UDP)[5]. When a multi-host broadcast was needed, UDP packets could be routed using IP Multicast towards a multicast address [29]. Applications wishing to receive this awareness information could, thus, "tune in" by joining the multicast session. Meta-information about session management, such as multicast address and port, is communicated through the standard Metaglue notifier or direct method calls. This was deemed appropriate since this traffic was limited to infrequent, small packet data.

Deciding how to encode audio and video streams was less straightforward. Criteria included quality, bandwidth required, encoding and decoding overhead, and availability of codecs. The last criterion was ultimately what determined how much implementation work was required, and thus was the highest priority. Computational efficiency of encoding and decoding were actually the second highest, since, for simplicity, the encoding or decoding would be done inside a Java virtual machine. Actual codec image quality versus bandwidth requirements were tertiary concerns due to the lack of need of high-quality video transmission in awareness applications.

The video encoding chosen is the ITU-T standard H.263 video codec, a modern version of the H.261 codec popularized by early videoconferencing systems such as the Polycom PictureTel. The H.263 codec was chosen over over MPEG-1 or MPEG-2 due to it being designed for low-bitrate communications, and for being supported by most videoconferencing applications on virtually every popular platform. Similarly, the GSM 06.10 is a low-bitrate, lossy audio codec that deemed adequate for compressing awareness audio streams. GSM 06.10 was originally designed for

[5]Unlike TCP, UDP packets are not retransmitted when dropped.

use in compressing speech for cellular phones in Europe, and is still widely used for this purpose today [49]. Optimization for low-bitrate communications seemed potentially important to allow mobile devices to receive transmissions via wireless transports such as bluetooth or WiFi 802.11b[6] Finally, an efficient Java implementation of both codecs was provided in the Java Media Framework (JMF), a library from Sun Microsystems designed for the purpose of capturing and manipulating media streams.

The JMF also facilitated encapsulation and transmission of the H.263 stream over the network. The RTSP (Real-Time Streaming Protocol), used widely across the Internet multicast backbone (MBONE), is an IETF-standard multimedia presentation control protocol designed for both unicast and multicast video and audio transmissions over IP networks [48]. RTSP supports Quality-of-Service metrics and reception quality monitoring through the RTCP (Real-Time Control Protocol), integrated into a session management mechanism for multicast media broadcasts. RTSP works with its transport protocol, RTP (Real Time Protocol), which are both directly supported by JMF, providing a simple means by which multi-stream broadcast sessions could be started, joined and managed. Unlike HTTP, which is transported over TCP, RTSP is UDP-based and leaves retransmission up to the higher-level application or media session manager.

4.3.3 O2Flow Architecture

The internal architecture of the O2Flow Agent is visible in Figure 4-2. As can be seen, the process of capture and transmission of awareness audiovisual streams is accomplished by using Java Media Framework's facilities for capturing media from low-level system devices, encoding, and encapsulating them for network transport.

JMF accomplishes data capture by controlling hardware capture devices such as microphones and cameras through native interfaces exposed through the underlying operating system. After JMF captures a frame of data, O2Flow passes the

[6] In retrospect, however, the rate at which wireless bandwidth is increasing makes this less of a concern.

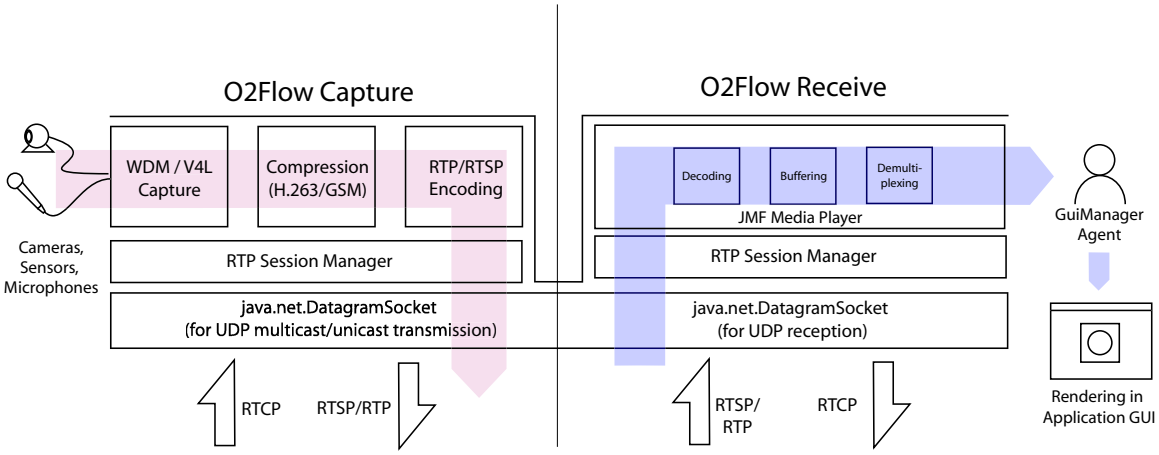


Figure 4-2: O2Flow Internal Architecture

data through several filters that compress and prepare the frame for transmission over the network. As described earlier, the H.263 and GSM codecs are compression filters that convert raw frames and PCM-encoded audio into low-bitrate data streams. After data compression, another filter is used to encode the data in an RTP-compatible format. Finally, the data gets passed to the RTP Session Manager, which manages the transmission over the network. O2Flow then either requests the Session Manager to create a new RTP session for the transport, or to add the stream to an existing multicast session already in progress. A multicast session can support any multiple concurrent streams originating from different sources, where all participants jointly receive all the streams in the session. The Session Manager then starts the stream, using the RTSP for broadcast meta-data, and RTP for the data payload. Clients who have joined the session periodically send out RTCP replies to each of the broadcasting sources, with information about their reception statistics. This informs the Session Manager about current network conditions, which allows it to potentially take corrective measures, such as such as by offering a more highly compressed (lower quality) stream in response to high network congestion. The rest of the details of data transmission are handled automatically by lower-level protocols, including IP multicast.

For receiving streams, O2Flow relies on the JMF Media Player object to perform

the necessary depacketization, decoding, buffering, and rendering. O2Flow places control of the Media Player's video canvas and graphical control components to the GUI Manager Agent, a Metaglue service responsible for laying out and composition of graphical components. Under normal circumstances, the GUI Manager queries the Metaglue Resource Manager to determine what display resources are available, and then transports the GUIs across machines to acquire a particular display resource. However, since the O2Flow agent GUI components must display video and audio, the GUI must exist on the same machine as an O2Flow agent instance. Thus, the O2Flow agent explicitly requests the GUI Manager to respect this constraint when laying out the display component.

4.4 Simple Perception Agents

To enable perceptual computing, Metaglue needed a means by which it could handle streams of perceptual video data. The Simple Perception Agents (SPA) were designed to fulfill this purpose.

4.4.1 Design Goals

The design goals of the SPA architecture can be described as follows:

1. *Simplicity and Flexibility* - The architecture should be simple and straightforward to implement, aiming first for basic perceptual applications, yet sufficiently flexible for future development of more advanced vision algorithms.

2. *Perceptual Capabilities On Demand* - Vision algorithms should be treated themselves as agents, or at least as dynamically loadable modules that can be spontaneously invoked to provide perceptual capabilities on demand.

3. *Scalability* - Finally, any vision system being designed, particularly in Java, should consider the computational capacity of such a system

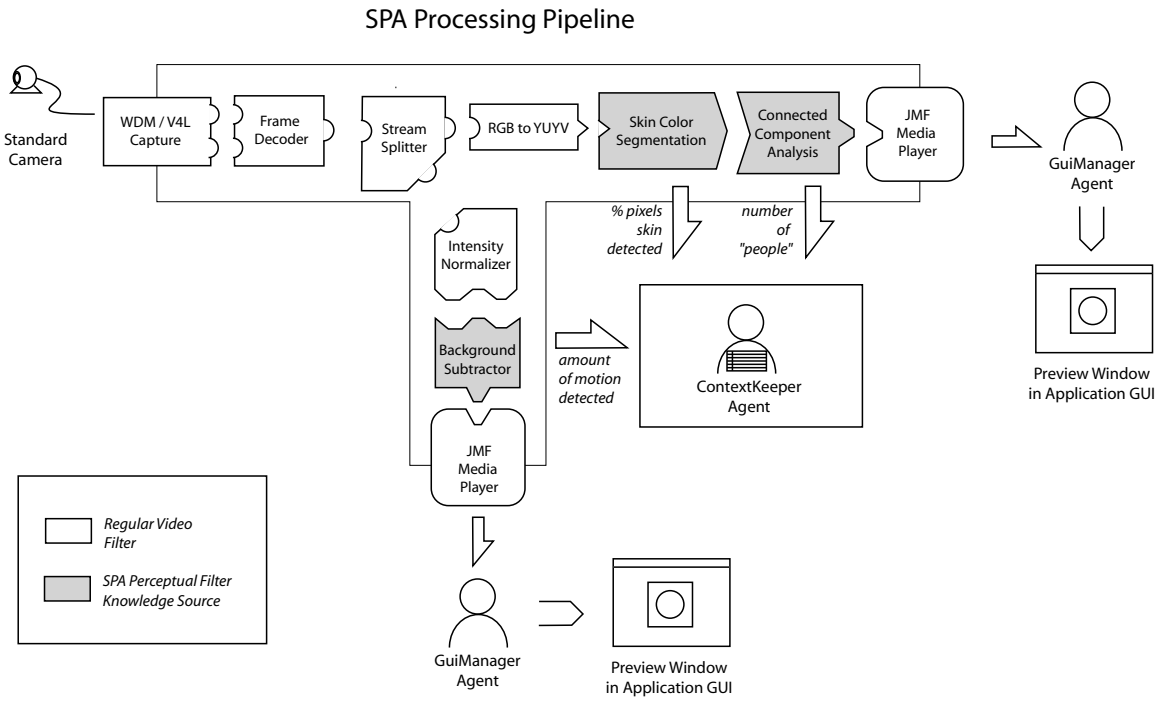


Figure 4-3: Simple Perception Agents

4.4.2 Architecture Description

As can be seen by comparing Figure 4-3 with the architecture of O2Flow, SPA strongly resembles the "capture" half of O2Flow. SPA, again, uses the Java Media Framework to gain access to capture devices, and passes the raw data obtained through a series of filters. However, instead of compressing and encoding this data and preparing it for transport over the network, these filters perform simple machine-vision algorithms on the data, publishing these results to the rest of the Metaglue system. This publishing is done through the perception-inferencing blackboard system, which will be presented in section 4.5.

The core of the SPA architecture consists of a processing pipeline. Frames grabbed from a capture device are inserted directly into the front of this pipeline. Each stage embodies a single logical processing operation, such as colorspace conversion or face detection, and passes the resulting data on to the remainder of the pipeline. A split may be inserted at any point in the pipeline to create two different parallel pipelines stemming from the same stage. Each branch off of a split

gets an identical copy of data coming into the split. Breaking up vision processing into pipeline stages in this manner fulfills modularity and allows for the expression of arbitrarily complex vision processing through construction of the branched data pipeline.

The vision pipeline is built dynamically at runtime by the SPA system. The system allows agents to manually specify how they want the pipeline constructed, which allows users to build custom processing pipelines exactly as they desire. However, SPA also supports automatic pipeline construction which pieces together a pipeline using information from each module about what it requires and produces. This allows the SPA to be treated more as a generic service that other agents can use to ask directly for perceptual capabilities, and helps fulfill the capability-on-demand objective described in the previous section.

Automatic pipeline construction is performed by working backwards from the desired end-goal. Each SPA pipeline component is designed to answer queries about the type of data input that it requires, and the output that it generates. Using this information, it is trivial to search the space of pipelines to generate a pipeline that bridges the source to the desired module. For example, if a face detection capability were desired, SPA first searches all modules for one that, for example, specifies that it could output the number of human faces it detected in a frame. It then queries the input requirements of that module, and searches, in turn, for a module that can satisfy these new requirements. This proceeds recursively, building up a potential pipeline, until the input requirements can be satisfied by the capture device itself, or no pipeline is found. In the latter case, the SPA system notifies the agent society that it was unable to fulfill the request.

4.4.3 Prototype Vision Algorithms

A few basic vision algorithms or *perceptual filters* have were implemented in the first version of the SPA. A description of these initial SPA perceptual filters follows:

1. *Difference/Motion detector* - This filter uses background-subtraction to deter-

mine how much a scene has changed from a known empty scene captured either periodically, or during calibration. This filter can also be used to determine the amount of motion in a scene by taking the absolute difference between pixels in successive frames.

2. *Skin detection* - Douglas Chai and King Ngan demonstrated a technique in [7] for quickly and easily isolating pixels in images that have a high likelihood of representing human skin. Their technique rests on their discovery that when images of various skin tones are analyzed in the YUV colorspace, the color balance components U and V remain consistent across ethnicities. Thus, their technique for skin segmentation involves simple thresholding of the U,V components. This simple technique was borrowed and integrated into the SPA for skin pixel segmentation.

3. *Low-pass filtering* - Also demonstrated by Chai and Ngan paper, various techniques of low-pass filtering segmented image data may be useful for certain other vision operations, such as finding coherent connected regions. In their paper, they introduce a low-pass filtering technique known as *density regularization* using simple erosion and dilation procedures. In the initial version of the implementation, the SPA low-pass filter performs simple averaging and thresholding instead of dilation.

While these methods are by no means representative of modern algorithms in computer vision, they nonetheless provide basic examples of how algorithms can be written and integrated into the SPA. In addition, while being extremely simple and limited, combining these prototype plug-ins yields pipelines that can detect when a person has approached or left the kiosk, as well as approximately how many people are standing near the kiosk at any time. Thus, these filters already provides interesting and potentially useful perceptual information that can be processed and handled by other kiosk software components.

4.4.4 Distributed SPA: Extending SPA for Distributed Vision

Machine vision algorithms can be computationally expensive, requiring extensive mathematical computation, and sometimes needing several passes over each pixel in each frame. Although, for simplicity of implementation, these SPA vision processors are implemented in Java, this implementation choice makes performance-related issues much graver. Most real-time machine vision systems today rely on optimizing code for specific machine architectures to allow execution at speeds several orders of magnitude faster than unoptimized code. Yet, this is not possible for interpreted Java bytecode.

One way to alleviate such a computational bottleneck is to embrace one of the tenets of ubiquitous-computing: to allow computational capacity, like any other ubiquitous computing service, to be treated as global resource flowing across physical machine boundaries. Distributed agent systems like Metaglue were designed to make this easier, by permitting agents to migrate between compute nodes, and by simplifying communication and coordination between nodes. Furthermore, an application such as machine vision lends itself to distributed computing because most vision algorithms feature substantial parallelism. This property permits vision algorithms to be decomposed into smaller steps, some of which must be done sequentially while others can be done in parallel[7].

By leveraging Metaglue's distributed agent facilities, the SPA may be able to support distributed vision processing to arbitrary granularity. This could be accomplished by modifying the SPA so that individual vision processors are embodied as separate agents that coordinate using the standard Metaglue communications mechanisms. Sharing frames or streams of frames between nodes could be simplified via a multicast mechanism resembling O2Flow, where any number of nodes can listen to the results posted from any previous nodes. The perceptual blackboard, described next, could facilitate the re-integration of this distributed processing into simple, coherent conclusions. Although such a distributed version of SPA has not

[7]For an early example of machine vision across a set of heterogeneous workstations, see the DeVIouS system: [22].

yet been realized, the design of SPA has been made to facilitate such an extension.

4.5 Perceptual Blackboard

One of the greatest challenges in ubiquitous computing software design has been determining how to coordinate all the components that constitute an IE. In particular, the input half of this problem consists of how to process the large quantity of potentially related, yet heterogeneous, contextual and perceptual data originating from a large number of sources. How exactly to merge data in a general, yet coherent and constructive, way is the topic of *perceptual* or *multi-modal data fusion*, an area of much active research today.

4.5.1 Design Goals

1. *Simplify application design* - The main design goal of the perceptual blackboard is to provide a convenient interface for end-user ubiquitous applications to access information related to context and perception. One way to do this is to choose to provide an event-driven interface to the application, just as conventional GUI toolkits do today.

2. *Diversity in models and inferencing engines* - Since perceptual fusion in kiosks is a general problem dependent on the particular application or task at hand, it would be dangerous to commit to a particular model or inferencing engine, such as neural networks or a rule-based system. Thus, the architecture should accommodate, yet be independent of, any specific inferencing algorithm or technique.

3. *Flexibility in knowledge representation* - Again, since the architecture must support a variety of applications and tasks, it should support flexibility in knowledge representation.

4. *Modular capability* - The system should be able to dynamically handle requests

for specific types of perceptual information by activating expert modules on an as-needed basis.

5. *Storage and retrieval of historical context* - Since historical context could prove useful for future inferences, this system should store and manage retrieval of historical contextual information.

4.5.2 Why a Blackboard?

The process of transforming raw perceptual inputs into data that is useful to applications is a problem that is ill-defined and potentially complex. As described in this section, blackboard architectures are well-suited for exploring such problems, where formal models individually fail, or where an appropriate model has not yet been realized. Furthermore, a simple mapping from blackboards to distributed agent systems make blackboard architectures even more appealing for Metaglue.

Blackboards form general-purpose knowledge-based system architectures that remain uncommitted to any particular formal model or inferencing technique. This gives blackboards a flexibility and neutrality not present in individual inferencing engines. Another useful property exhibited by blackboards is their inherent modularity. All sources of information or expertise are encapsulated into *knowledge sources*, which are individually independent entities, but may be made to cooperate and communicate across the blackboard. Since knowledge sources work independently, they can be added, removed, or replaced at will without breaking the system.

Since a blackboard knowledge source can be made to embody any formal model or inferencing algorithm, any blackboard architecture is at least as capable as its constituent components alone. However, according to Daniel Corkill, blackboard systems are, in fact, inherently more powerful because inferencing can be accomplished across heterogenous graphical models. This inferencing may be done in an *opportunistic* manner[8], or in a goal-directed fashion. This property, Corkill de-

[8]Corkill defines *opportunistic* as meaning having the control capability to perform the most ap-

scribes, makes blackboards much better-suited for ill-defined problem areas [11]. The first blackboard system, Hearsay II, was itself built to explore a new problem domain at the time, speech recognition and natural language understanding [43]. Penny Nii has also demonstrated that blackboard architectures are able to conduct both data-directed and goal-directed inference simultaneously, or to switch between the two through merely by manipulating the scheduling mechanism [36]. This ability to build out from *islands of confidence*, hypotheses the system believes is most likely to be correct, enables the type of opportunistic inferencing that distinguishes blackboards from many other inferencing systems in AI, which are usually either goal-directed or data-driven.

Blackboards have also been used to facilitate coordination and cooperation for problem solving in distributed agent systems. In such systems, the blackboard itself may not exist in any one location; instead, relevant pieces of it may exist inside each particular agent in the system. In such distributed blackboard systems, synchronization is maintained through message-passing, publish-subscribe or a broadcast mechanism, without any loss of functionality or generality [16]. As will be described in the next section, since blackboard architectures are functionally equivalent regardless of whether they are distributed or centralized, the first Metaglue blackboard prototype, the ContextKeeper, was designed as a service embodied in a single Metaglue agent.

4.5.3 The ContextKeeper: a Blackboard Prototype for Metaglue

The remaining parts of this section are dedicated to describing the ContextKeeper, the first-generation prototype blackboard for Metaglue. This blackboard is inspired by the Simple Blackboard System presented by Corkill in [11], but is adapted using Metaglue's object-oriented (Java) agent design methodology.

propriate problem-solving action at each step in the inference process.

Blackboard Representation

In ContextKeeper, knowledge sources are embodied in Metaglue agents. These knowledge source agents, however, are special in that they must adhere to a strict social contract with the ContextKeeper Agent, which manages the storage, retrieval, and access control of all blackboard knowledge. This social contact permits the ContextKeeper to arbitrate the direction of inference in the system, by scheduling and choosing which agents get to contribute facts. As described earlier, Nii and Corkill describe scheduler control as a critical component for directing inference.

The ContextKeeper blackboard itself stores all incoming knowledge in a Metaglue hashtable, which is backed for persistence by a database. This knowledge can take the form of any Java serializable object that conforms to the *ContextData* interface.[9] This interface contains methods required by the blackboard to organize and manage knowledge objects. These methods include accessors for the time of the context data object's creation, the agent identifier of its creator, and a *type specifier*. This last piece of information designates the partition, or layer, of the blackboard to which the data belongs. The ContextKeeper supports an arbitrary number of such layers, used to reduce search during retrieval. These layers can represent levels of abstraction, such as in HearSay II, or any other logical partitioning scheme.

Truth Maintenance and Knowledge Management

Within each layer of the blackboard, the ContextKeeper collects knowledge objects into a simple list, ordered by time of creation. Once a piece of knowledge has been added, it is considered immutable and cannot be retracted or modified by any external agents. However, it may be *deprecated* if another assertion gets added that contradicts the original assertion. Thus, the notion of "truth" in the blackboard is represented by the *frontier* of the blackboard; that is, every assertion that has not been deprecated is considered "true". By deprecating, rather than replacing, obsolete contextual data, knowledge sources can continue to have access to historical

[9] a simple implementation of the ContextData interface which can be extended is also provided, called *BasicContextData*

data for inferencing based on temporal patterns[10].

The ContextKeeper grants all Metaglue agents access to knowledge on the blackboard through a query interface that supports a number of different types of search requests. Most of these requests involve finding the most recent piece of knowledge that fits a given search criteria. For these requests, the algorithms start at the end of the list (where the most current items are located) and perform a linear reverse search examining each entry until the query is satisfied.

Unfortunately, preserving all historical context data means that the blackboard can amass a virtually unbounded amount of knowledge. This means that the ContextKeeper agent would continue to grow until it exhausted all available physical memory, and then cease to function. Searches through such an enormous database could take an unacceptable amount of time. Thus, the ContextKeeper needs a means of *knowledge retirement* to enforce an upper bound on the amount of memory and time that searches may take. However, retirement must be done carefully, so as to not counteract the purpose of keeping historical information in the first place, and must make sure to not adversely affect knowledge sources' ability to function.

The ContextKeeper supports fine-grained knowledge retirement through the use of dynamically interchangeable *retirement policies*. An example of such a retirement policy would be the N-first-in-first-out policy, which enforces a limit of N distinct knowledge elements of a particular type by removing the oldest such element when a new one is added. Another example would be the temporal-resolution-reduction policy, which lowers the frequency of older knowledge elements by pruning old, temporally proximate elements. Such a policy could be useful for storing data such as the weather, where keeping track of every half-hour weather update for a day two weeks ago provides virtually no additional benefit over knowing the approximate nature of the weather for the whole day. Custom retirement policies can be defined simply by implementing the *ContextRetirementPolicy* interface, and by instructing the ContextKeeper to respect this new policy for a particular type of knowledge

[10]Markov chains or hidden Markov models are examples of types of knowledge source that use temporal patterns to synthesize current hypotheses

data.

Knowledge Sources

The interaction between the ContextKeeper and the knowledge sources is predominantly a *publish-subscribe* mechanism. When a knowledge source is enabled, it first registers itself with the ContextKeeper. During this registration process, the knowledge source asks the ContextKeeper for a subscription to notifications of certain types of knowledge when they get added to the blackboard. The ContextKeeper performs a callback to the knowledge source when matching knowledge is added to the blackboard.

After the ContextKeeper has finished informing everyone of the new piece of knowledge, it queries each of the knowledge sources to determine if any of them have *triggered*. Triggering indicates that the knowledge source has computed new knowledge that it wishes to contribute to the blackboard. If this is the case, control is handed over to the ContextKeeper's scheduler,which then decides which knowledge source gets to actually *fire*. Here, firing means that the knowledge source gets to contribute its new information to the blackboard. When this new piece of knowledge has, at last, been added to the blackboard, knowledge sources are informed of the update, and the process repeats again.

Scalability Concerns

Since access to the blackboard must be performed serially (to prevent concurrency-related race conditions that may cause erroneous behavior), there is a significant risk that the ContextKeeper will form a performance bottleneck in the Ki/o architecture. In this design, it is clear that the ContextKeeper's ability to fulfill search requests, dispatch, trigger, and fire the appropriate knowledge sources is going to be the limiting factor for the speed of the blackboard.

Unfortunately, optimization of this has proven difficult. First, in order to keep the triggering conditions as general as possible, the current design has resorted

to explicitly querying each of the knowledge sources whether it has triggered every time a new piece of knowledge has been added to the system. Historically, rule-chaining systems facing the same problem have been able to optimize trigger-checking by using a RETE network, which groups identical triggering conditions across rules together, so that they would only have to be checked once [17]. However, knowledge sources in this system could potentially have complex, mathematical or probabilistic triggering conditions not easily expressible or decomposable as a logical combination of predicates. As a result, the current design encapsulates trigger checking within a method in the knowledge source.

Fortunately, two simple policies regarding the design of knowledge sources can greatly help keep the ContextKeeper free:

1. *Knowledge Caching* - Knowledge sources should minimize the degree to which they query the blackboard through its query interface. This can be done by caching new knowledge notifications even if they do not immediately result in a trigger. That way, when knowledge later arrives, it may be combined with cached knowledge to permit the knowledge source to trigger and fire without having to query the blackboard.

2. *Freeing the ContextKeeper thread* - As mentioned earlier, all calculations involved in trigger-checking and firing should be performed on the knowledge source's own thread instead of in the ContextKeeper's thread. This implies that the methods entered by the ContextKeeper should merely access shared variables that have already been set by the knowledge source's internal thread. These measures permit the ContextKeeper's thread to peek in and drop or grab appropriate information and run to the next request.

Further optimizations to the ContextKeeper will be postponed for future work.

Prototype Knowledge Sources

The first application to use the ContextKeeper is the Ki/o Web Information Portal, an intelligent web browser that attempts to present visitors pages that they might

Visual perception observations:
1. People are passing by
2. One or more people are standing in front of the kiosk
3. Nobody is nearby

GUI events:
1. User has touched the screen
2. User has requested a master navigation button
3. User has requested a hyperlink on a web page

External knowledge:
1. A new weather report has been received
2. A new news story has been posted on BBC
3. A new news story has been posted on CNN

Inferred hypotheses:
1. A person is observing the kiosk.
2. A person is actively interacting with the kiosk
2. The person who was in front of the kiosk has left

Table 4.1: Perceptual events generated by the ContextKeeper

be interested in seeing. In order to determine how it should behave and what pages to display, the web display application relies on the ContextKeeper to feed it contextual clues.

The initial set of knowledge sources developed for this application post simple observations about world. A complete list of the types of observations that these initial knowledge sources generate is illustrated in Table 4.1.

The first collection of visual perception observations are posted by various perceptual filters among the SPA agents. For example, the observation "People are passing by" relies on the SPA motion detector filter, which measures and reports differences between successive captured frames. The second and third observations are fed by the skin color segmentation filter, described earlier, which detects and isolates pixels that resemble the color of human skin.

The *GUI events* describe any interaction with the touchscreen. These events are

mapped to standard mouse events, such as pointing, clicking and dragging.

External knowledge is posted by miscellaneous Metaglue agents that retrieve knowledge from a variety of external sources. The initial set of agents retrieve weather and news information by connecting to various Web services. These agents are responsible for performing initial parsing, processing, and filtering of such news and weather data so that it is in a representation that is easily interpretable by Metaglue agents.

The *inferred hypotheses* are knowledge sources that read the events that other sources have posted on the blackboard, and combine them to produce speculative interpretations as to what actually occurred. As a simple example, if the SPA perception filters detect skin coloring in front of the kiosk, and recent touchscreen events are posted, a knowledge source will make the (obvious) hypothesis that someone is currently standing in front of, and interacting with, the kiosk. While this may initially seem like a trivial and useless inference, adding this hypothesis frees applications by allowing them to subscribe to this higher-level interpretation instead of directly to the lower-level events. This layer of abstraction allows the lower-level events to be modified without the application being adversely affected.

4.6 Putting it All Together: Applications

The previous sections discussed details of various new components introduced into the Metaglue system for the purpose of building perceptual, contextually aware Ki/o applications. Since the success of these designs ultimately depends on how well the components actually serve their purposes, the components of the Ki/o architecture have been tested in a real-world setting.

This section will illustrate the first set of Ki/o applications, some of which have already been deployed on the prototype kiosks in the laboratory, and others of which are still early works-in-progress. Although these applications are largely "toy" applications, and do not exploit all of the functionality of Metaglue or the Ki/o architecture, they do provide a starting point for designing richer applications

in the future.

4.6.1 Web Information Interface

Since one of the core objectives of the Ki/o kiosk prototype is to facilitate information access and dissemination throughout the laboratory, the first application designed for the Ki/o kiosk was chosen to be an information access portal. Instead of designing a format and renderer specific for the Ki/o display, the Web seemed like a logical starting point due to its ubiquity, visibility and accessibility. Thus, as described previously and as visible in Figure 4-4, this application consists of a relatively conventional graphical Web browser user interface, with navigation and status bars at the top. Unlike a regular Web browser, however, behind the scenes, this application relies on perceptual and contextual information to determine what content to display.

Design

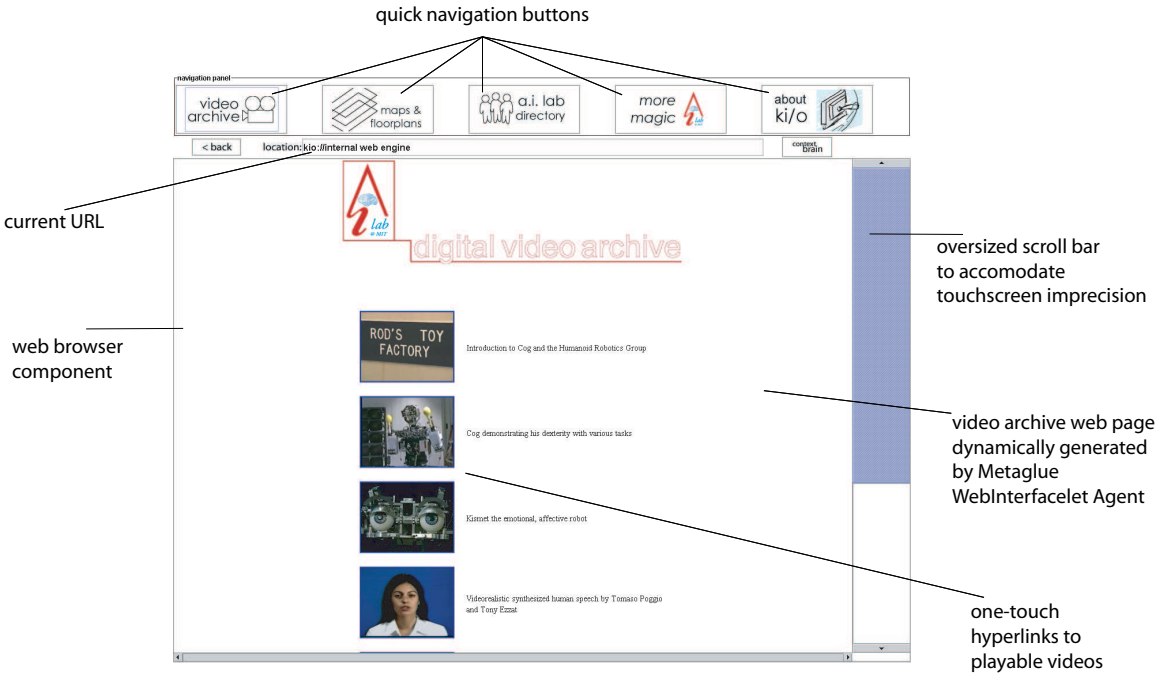
The design of the Web Information application can be logically divided in three simple pieces:

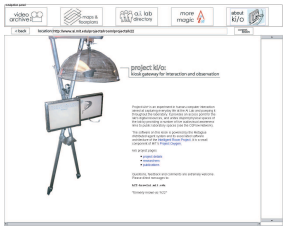
- *User Interaction Component* - This is analogous to the the UI of a traditional application, including all on screen GUI widgets. The main GUI widget in this application is a large HTML rendering panel.[11] This layer packages all user-generated GUI events, such as activation of hyperlinks, into ContextData objects that it feeds the ContextKeeper blackboard.

- *HTML Generator Agent(s)* - The HTML Generator agents format particular types of data into HTML on demand for display by the UI. This component can be thought of as an encapsulation of Java servlets as Metaglue agents.

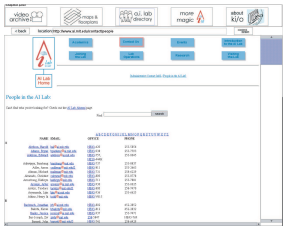
- *Supervisor* - The supervisor component holds the application together, by driving the UI and HTML generators. The supervisor schedules content to be dis-

[11] For the initial prototype, a Java Swing JEditorPane was used in place of a full web browser. Plans for replacing it with a more standards-complete HTML renderer are underway.





Ki/o project information guide



AI Laboratory personnel directory

MoreMagic computing help page

Figure 4-4: Screenshots of the Web Information Display, v. 1.0

played, and coordinates with the ContextKeeper to determine when, and what to display.

4.6.2 Supervisor

The Supervisor component forms the core application logic, and has a number of responsibilities. In its simplest incarnation, the Supervisor acts as follows: First, it watches the ContextKeeper for urgent messages to display. In an emergency or urgent situation, any agent posting such a message would cause the Supervisor agent to immediately pass the information through an appropriate HTML Generator agent, and then to the HTML renderer for display. Otherwise, the supervisor watches for other sorts of perceptual and contextual events. If a hypothesis of high certainty is posted that nobody is nearby, the supervisor starts cycling the display, either choosing content at random, or recently updated information, such as news stories, that are likely to attract passers-by.

Alternatively, more sophisticated content choosing strategies should be simple to implement, particularly as the supervisor has access to all the contextual information on the blackboard. Experimentation with such strategies is left for future work.

4.6.3 AI Lab Video Database

In addition to Web-formattable content, a number of historical AI Lab movies were available to be displayed by the Ki/o. Unfortunately, the simple HTML renderer chosen is implemented in Java, and does not support Netscape plugins or ActiveX controls. Thus the standard mechanisms for displaying video on the web was inappropriate. To get around this limitation, HTML Generators were used to render a custom web page for selecting the appropriate video, which contained links to special Metaglue servlets. As these servlets could then be used as a bridge back into Metaglue from a Web page, it was then a simple matter to command Metaglue to display the appropriate video. In this case, for displaying videos, Metaglue used its

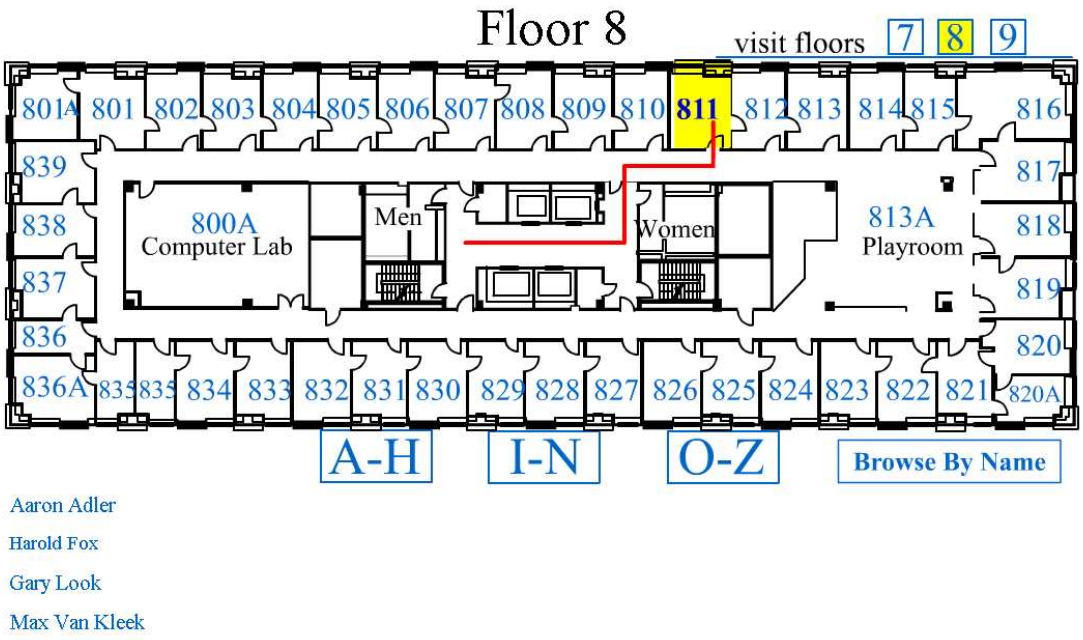


Figure 4-5: Screenshot of The Ki/o Interactive Floorplan, v. 1.0

internal GUI Manager agent to invoke a Java interface to Apple's QuickTime player, which performed the actual decoding and playback.

4.6.4 The Interactive Floorplan

The first Ki/o prototype was deployed in an elevator lobby of the AI Laboratory. At this location, many people construed the Ki/o prototype to provide floor information service, perhaps from prior experiences with kiosks in modern hotels and skyscrapers. It thus became quickly apparent that a map of the floor layout, with a directory of lab occupants would be useful and popular.

Version 1: Simple Interactive Floorplan

The first version of the floorplan application, developed by Ki/o by Justin Maynard and Mark Pearrow, can be seen in Figure 4-5. It consists of a two-dimensional representation of the floor from above, and allows users to touch any space they

want to know more about. The application then draws a path indicating a route from the kiosk to the destination, queries the AI lab personnel database to find the room's occupants, and displays the results. Each office and lab space is labeled with its official room number.

If the user was looking for a specific person without knowing their office, they could look up the appropriate location via the person's last name. Once the name is displayed, clicking on the name displays a path from the kiosk to the appropriate person's office.

While this application served its purpose well for the 8th floor, unfortunately, it was limited to the top three floors of the laboratory. After its initial deployment, it quickly became apparent that similar floorplans for the Laboratory of Computer Science, occupying the lower floors of the building, as well as the AI Lab's remote annex, were absolutely necessary.

Furthermore, instead of modeling the physical spaces within the floorplan program, each floorplan was treated as a simple bitmap. As a result, the routes that were displayed had to be hard-coded, as were office labels and the kiosk's location.

Seeking a more flexible solution, the design for Version 2 of the Floorplan application was devised.

Version 2: 3D Awareness Floorplan

To determine how to model the physical layout of the building in a flexible and efficient manner, the Computer Graphics Group in the LCS was consulted. Coincidentally, Jason Bell, a master's student with the Computer Graphics Group, had been concurrently developing an application programming interface (API) to simplify access and representation of internal building geometry data. This floorplan application seemed a perfect trial application for his API, and thus a collaboration was formed.[12]

Through the rich representation of space provided by the Computer Graphics Group's data and Bell's API, it was possible to model both the floors inside the

[12]For specific details about this API, refer to [1].

laboratory with respect to one another, as well as other remote buildings on the MIT campus as well. Thus, it became possible to model satellite offices outside the AI Laboratory, as well as to other related laboratories around campus.

For version 2 of the floorplan, the following design goals were established:

1. *3D visualization* - The display should render the floors of the laboratory as a single, integrated 3D model reflecting its physical layout to allow users to easily see spatial relationships between interior locations.

2. *Clear, update-able space labels* - Like version 1.0, all rooms should be labeled with its name and its occupants. To avoid 3D clutter, the display should attempt to dynamically expand textual labels upon user selection.

3. *Simple, user-intuitive navigation* - 3D views are frequently difficult to navigate using current today's 2D input devices. However, since it is critical that this application be intuitive and easy-to-use for first-time users, by default the application should provide constrained navigation to a number of easily-selectable pre-set viewing angles, and only provide free-form navigation for advanced users.

4. *Intelligent Route finding* - A route-finding algorithm should be developed which can automatically compute paths between arbitrary landmarks within the building, including those on different floors and in remote annexes. This route-finder should be able to additionally provide users with a textual description of directions for clarification.

5. *Situated icons displaying activity in the Lab* - In addition to providing static information about the laboratory's physical layout and its floor directories, the floorplan provides a convenient way of conveying special events in progress within the laboratory, such as lectures and seminars, at-a-glance. This awareness information, which might be retrieved from the ContextKeeper or external sources such as lab event calendars, may appear as icons in the model corresponding to the respective location in the lab.

This 3D floorplan will serve an important role in helping laboratory members find one another when the AI and LCS are reorganized and transition to the new space in the Stata Center. Additionally, visitors to the Stata Center will most likely find the floorplans helpful in navigating the radical arrangement of spaces in the new building.

Current status

At the time of this writing, implementation of the 3D floorplan is still in an early stage of development. An initial prototype developed by Bell to demonstrate Java3D rendering of floorplan data acquired through his API is visible in Figure 4-6. A full prototype is slated for completion by Summer 2003.

4.6.5 Competitive Crossword Puzzle Application

Description

A third Ki/o application is an experiment towards inspiring competitive, friendly social interaction among members of the AI Lab. The application is a multi-player competitive crossword puzzle game, which allows any number of people to simultaneously try to fill in the puzzle from any AIRE space within the laboratory.

This game can be played in a synchronous or asynchronous fashion. In synchronous mode, everyone accessing the puzzle sees the exact same view, including guesses made by other people. When the board is live, it displays a list of other people who are currently participating, and updates the board as people fill in their guesses. If multiple people enter a guess simultaneously, an arbitrator agent checks timestamps and determines which person was first, and gives that person credit. Since guesses are not verified by the system until the game has ended, the system allows people to *clobber* another person's guess with their own. However, if the original person's guess was indeed correct, the victim is rewarded and the clobberer's score is reduced.

When a board is filled or the time is up, a game is considered over. Players'

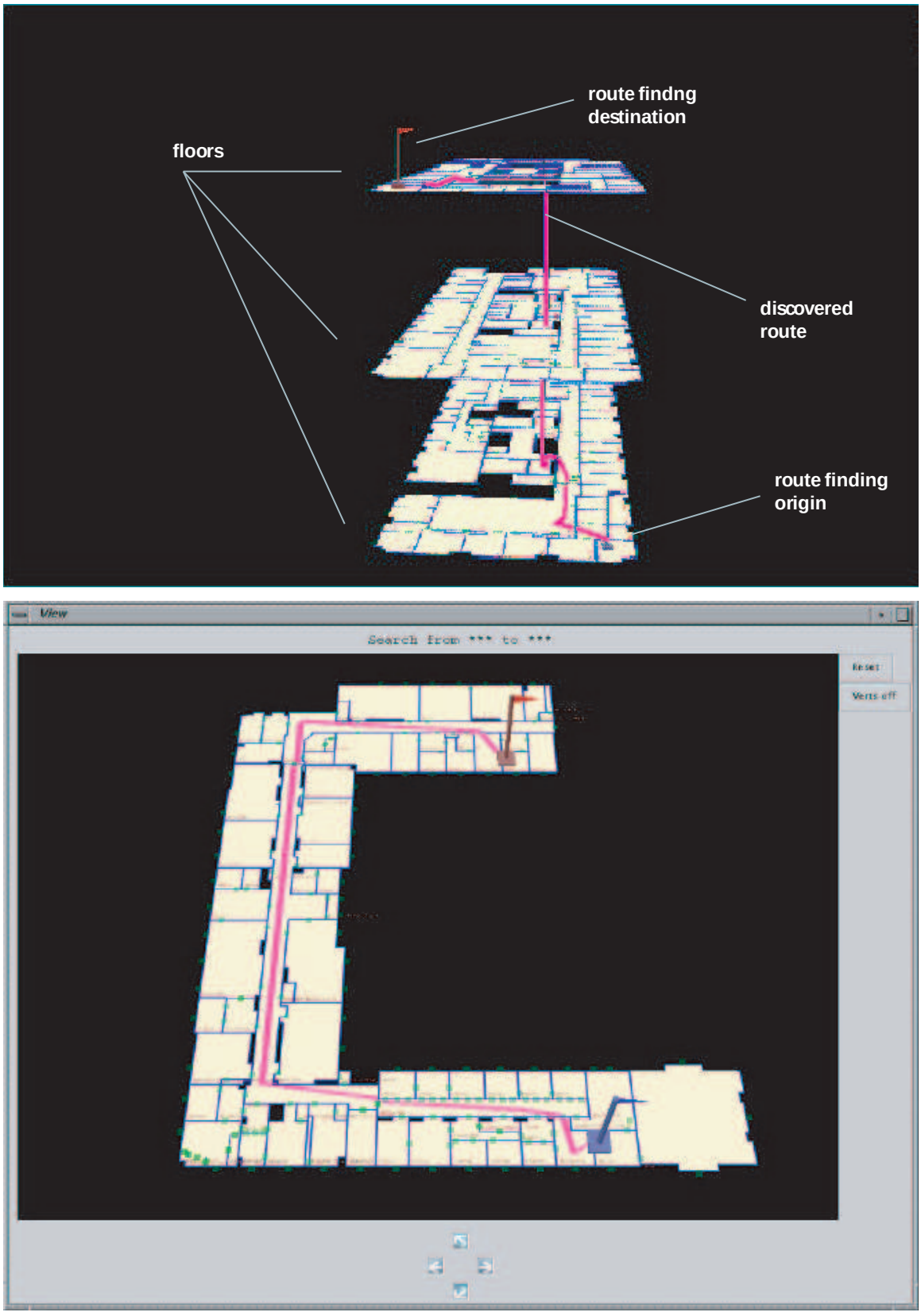


Figure 4-6: Annotated screenshots of early 3D Floorplan Prototype (courtesy of Jason Bell).

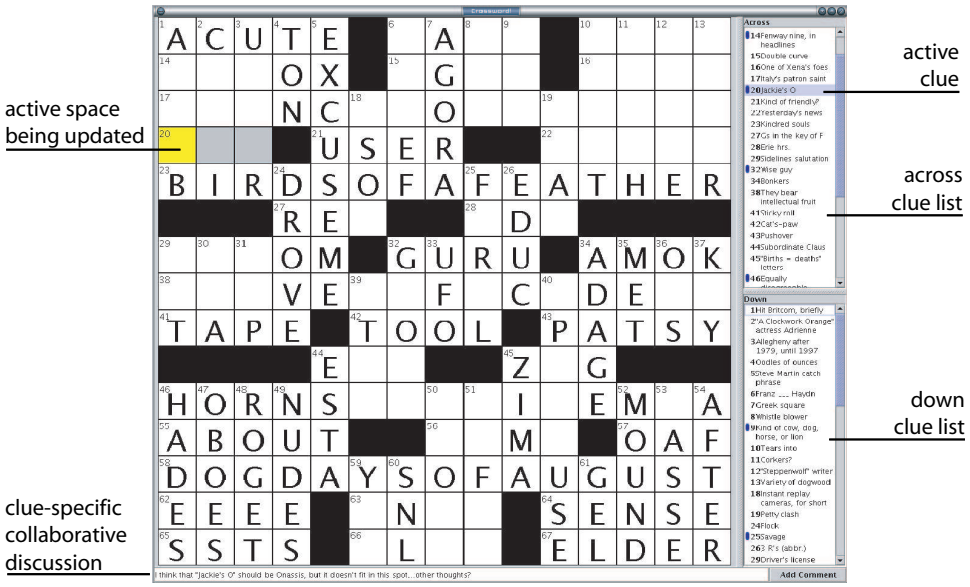


Figure 4-7: Screenshots of early collaborative crossword prototype (courtesy of Tyler Horton).

scores are calculated in proportion to the number of correct words they have in the puzzle, minus the number of incorrect clobbers they performed. Players are then grouped into "teams" by research group, and overall statistics are collected and displayed on a daily or weekly basis. New games are started by the system at random times throughout the day.

To maximize the social effects of the game, each person's guesses are labeled with the person who made that guess. Players can leave comments on clues to help (or even distract) other participants. The list of current participants can be used to initiate a two-way communications link with a participant through O2Flow audiovisual links, or the MetaChat, Metaglue's instant messenger service.

In asynchronous mode, identical puzzles are issued to different locations, but the puzzle is not updated between sites. Instead, statistics about each team's progress is displayed abstractly, as a graph, to encourage competition and participation. Experiments will be conducted to determine which version of the game is more popular.

Implementation Status

This application is currently undergoing active development with the help of Tyler Horton, an undergraduate researcher in the AIRE group. A screenshot of the current prototype he developed is visible in Figure 4-7.

4.7 Conclusion

This chapter described the Ki/o software architecture as it currently stands, consisting of a wide variety of components responsible for providing basic awareness, media stream handling, perception, inferencing, and context management capabilities. However, support for many of the capabilities required to achieve the goals outlined at the beginning of the chapter, such as multi-modal interaction and seamless user state migration, do not yet exist. These are items of current active research, and will be discussed in the next chapter as future work.

Chapter 5

Future Work

5.1 Opportunistic Meeting Capture

One of the initial motivations for the Ki/o project was to see how intelligent environments in public gathering places could help capture and bolster new ideas as they are informally spawned or communicated. To study this, designs have been devised for an Opportunistic Meeting Capture application to detect, record, and index informal gatherings in front of Ki/o kiosks, and then later provide retrieval access to people who participated in the meeting. This application would also provide a whiteboard-like interface with which the gathered participants can scribble, manipulate notes on the touchscreen, and link to external documents and web pages. When participants later regrouped at the same or another IE, the meeting footage from the prior interaction with relevant notes would be automatically retrieved and displayed for easy manipulation.

Unfortunately, realizing this application is challenging at multiple levels, and as a result, only initial steps have been achieved. One initial effort has been made to determine how to best capture and synchronize all data coming into a system during a meeting, including touchscreen events, whiteboard writing, audio, and video input. All of these channels can potentially be used to provide information that can be to aid in later automatic retrieval. Harold Fox, a graduate researcher in the AIRE group, is currently attempting to extract speech from audio recorded

in multi-person meetings and use this to deduce the topics of discussion. Video processing can aid with determining who the meeting participants are, as well as to help indicate who is speaking at any moment.

A prototype of the meeting capture application is currently targeted for summer 2003.

5.2 Speech and Natural Language Interfaces

One approach to designing more natural computer interfaces is to model computer-human interaction after how humans interact with one another. This research has led to the development of *natural language* (NL), *dialog-based* interfaces that are capable of holding conversations with the user in the user's native language. Since these NL-based dialog systems can potentially eliminate all communication barriers between computers and humans, they have become viewed as a fundamental capability needed towards achieving the ubiquitous computing vision of allowing computers to enter and interact with the natural human environment.

5.2.1 Speech at the Kiosk

Natural-language dialog could improve interaction with Ki/o kiosks in a large number of ways. Since users could directly express what they want to the kiosk, interaction speed and efficiency would increase. By combining speech with other interaction channels, such as the touchscreen, users have dramatically more freedom with the way can they express their desires to the kiosk. This benefits of this sort of *multi-modal* interaction has been demonstrated in a number of prototype systems already [28][56][38].

5.2.2 Galaxy

The MIT Spoken Language Systems (SLS) group[1] has been devoted towards building a robust and feature-rich natural language dialog framework. This system, called *Galaxy* [50], was originally designed for speech-based applications over the telephone. The group has developed and deployed a number of such applications, including *Jupiter*, a weather information system, and *Mercury* and *Pegasus*, flight scheduling and status systems.

Although all of the SLS group's deployed applications have been telephone-based, a number of features of Galaxy make it a good choice for being used in a kiosk-like setting. Since telephone-based systems must be prepared to understand virtually anyone who calls, Galaxy has been built to be speaker-independent, and robust against line noise and speech impediments. Ki/o kiosks feature similar requirements, since they too, must be prepared for anyone to approach. Since Ki/o applications treat the user's attention as its most precious resource, speech on Ki/o must not require training or the user to wear any special device, such as a close-talking microphone. Galaxy already meets these requirements.

5.2.3 Research Direction

Due to the possibly great advantages towards integrating Galaxy with the Ki/o architecture, the two research groups have decided to pursue a collaboration with this goal in mind. This integration creates a number of challenges and design questions at both low, software architecture level, through the high, HCI level. These issues are outlined here.

1. *Interfacing Galaxy and Metaglue* - Galaxy and Metaglue have orthogonal architectures that are not directly compatible. Reconciling Galaxy's programmable-hub architecture with Metaglue's autonomous agent *scatterbrain* architecture requires a consideration of how these two systems should interface with one another. A high degree of integration could require substantial work for both

[1] See the SLS group's web site at http://www.sls.lcs.mit.edu/

architectures, while simply connecting the architectures may make it difficult and inelegant for use.

2. *Representing NL-dialog in Ki/o applications* - A question surrounds how to represent NL-dialog when designing ubiquitous applications. Galaxy supports spoken applications that are *mixed-initiative*, which means that neither the user and the computer assume a dominant role in a conversation. A goal of Ki/o would be to achieving this sort of collaborative, flexible interaction across all UI modalities[2].

3. *Multi-modal dialog and discourse modeling and management* - Currently, Galaxy's dialog and discourse management systems have been designed to handle context maintenance and inheritance for supporting anaphora, ellipsis and deictic expression resolution for speech-only interactions. Questions remain as to how the methods currently employed generalize to the multi-modal domain.

4. *HCI issues* - A number of new HCI issues crop up when the kiosk becomes capable of holding NL-dialogs with users. Most of these issues surround allowing the user to detect and recover from misinterpretations of what the user has said.

Early progress has been made in each of these major areas. First, there has been substantial progress by Nicholas Hanssens, a member of the AIRE group, to solve the Metaglue-Galaxy integration problem. He describes his solution in his thesis [20]. The SLS group, in conjunction with the Vision Interface Project (VIP), has very recently designed a prototype Galaxy Multi-Modal Server that tracks hand gestures with a camera, and ties it into the dialog model for resolving deictic references. Finally, discussions of potential solutions to the HCI issues surround the development of a dialog visualization application which tracks the progress of con-

[2]For a more detailed definition of mixed-initiative computing and various approaches that have been investigated, see [21]

versations and permits users to easily view how their utterances are being construed by the system.

5.3 Protecting User Privacy

This paper proposes that networked, perceptive intelligent environments could soon fill all public spaces in the modern workplace. With such a proposal, it would be negligent to overlook the possible dramatic social implications that such a system could have. Specifically, having machines capable of perception, and constantly collecting and processing observations presents a huge potential violation of personal privacy, as well as possibly the security of the workgroup at large.

5.3.1 Designing for Privacy in Ubiquitous Computing Systems

One of the primary open questions of much debate in ubiquitous computing research surrounds how to preserve and respect personal privacy. One of the greatest difficulties with solving this debate surrounds defining privacy itself, despite nearly everyone having an intuitive idea of what it is. Through repeated attempts, it has started to become apparent that any single attempt at defining privacy would sooner or later fall obsolete. In their seminal paper, "Design for Privacy in Ubiquitous Computing Environments", Victoria Bellotti and Abigail Sellen of Xerox EuroPARC defines privacy as a personalized notion influenced by one's culture and environment:

> Any definition of privacy cannot be static. With the introduction of new technology, patterns of use and social norms develop around it and what is deemed "acceptable" behavior is subject to change. Naturally evolving social practices may interact with organizational policies for correct usage (...) in addition, people are more prepared to accept potentially invasive technology if they consider that its benefits outweigh potential risks. In recognition of these facts, we take privacy to be a personal notion defined in response to culturally determined expectations and what

one perceives about one's environment [2].

This view of privacy proposes that it is a subjective notion dependent on individual preferences and situational context. Furthermore, it proposes that if the overall utility of a system is high, the established social norms may slowly accept it. Bellotti and Sellen have thus made it apparent that designing ubiquitous computing systems to protect individual privacy is highly a subjective process that depends on the setting, who is affected, and what the system does for users.

The Bellotti and Sellen studies were conducted in their HCI research laboratory at EuroPARC surrounding a video-based awareness and communications tool known as RAVE. The social environment of EuroPARC was similar to that of most small, close-knit research labs, where members place considerable trust with one another within their immediate organization. Bellotti and Sellen studied how they could design their system within this social setting to preserve and enhance this trust, and to ensure that their system did not break down the social norms of their workplace.

In the process of trying to reconcile RAVE with their laboratory's social disapprobation of capturing and recording video, Bellotti and Sellen devised design criteria for making ubiquitous computing systems that capture user data more socially acceptable. This is done by empowering the systems users by giving them immediate, clear, feedback and control over the *capturing*, *construction*, *accessibility*, and the intended *purpose* of data collection [2].

Unlike social, political, or cultural solutions to solving the problem of personal privacy, Bellotti and Sellen's solution is a simpler, technical and HCI design-focused approach that is easier to implement and evaluate. Furthermore, their case study with RAVE greatly resembles and may be directly applied to Ki/o awareness monitoring systems such as O2Flow and SPA.

5.3.2 Privacy in captured video data

Greater control over video streams in video links may be achieved through techniques in computer vision and video processing. Simple techniques might involve users dynamically controlling visual and temporal resolution of captured awareness data, by modifying the frame rate and frame resolution. Providing constant feedback of the degree of visibility is essential, and can be done by always providing a mirror or self-view of video.

New, recent advances in computer vision provide exciting new possibilities for allowing users to control their visibility. By training a system with a model of the user's face using *principal component analysis* under "socially correct", normal, or typical conditions, the system can later perform *eigenspace filtering* to morph live captured images back to their more "socially correct" counterparts. Example would be if a user wants to convey his or her presence without being properly groomed, the eigenspace filter can reconstruct the image while preserving other properties of the user's face and presence [12].

Machine vision can also enable the enforcement of certain social norms such as reciprocal visibility, such as only transmitting video if the other party is, likewise, visible. Other aspects of user control enabled by vision include being able to allow the user to control how much of the user's background and surrounding environment is preserved or removed. The *Reflection of Presence* videoconferencing system has demonstrated a way to entirely segment meeting participants from their respective office backgrounds, and to co-locate them against a shared, virtual background.

Although ensuring absolute security against the abuse of personal user data captured in ubiquitous computing settings may still be technically impossible today, techniques such as these are encouraging indicators of how users may be able to exercise a fine degree of control over the information they divulge in any ubiquitous computing setting.

5.3.3 Big Brother versus the Transparent Society

In many ways, intelligent kiosks observing and recording people from every corner of the workplace bears great resemblance to the viewscreens depicted in George Orwell's novel *1984*, which the totalitarian dictator Big Brother used to observe and control every aspect of the citizens' lives [37]. Orwell's *1984* presented a useful depiction the perils of how surveillance technologies could be abused under hostile or malicious intent. However, at the same time, the novel has so dramatically shaped the public perception of surveillance-or sensing-related technologies that a certain degree of paranoia, even panic, has become widespread.

David Brin, a more recently popular science fiction writer, has sought to help dispel the notion that surveillance and sensing-related technologies such as ubiquitous computing imply a loss of civil liberties and personal freedom. In his book, *The Transparent Society: Will Technology Force Us to Choose Between Privacy and Freedom?*, Brin describes how surveillance could bring about an *open society*, where cameras and surveillance technology would allow everyday citizens and public authorities alike to keep a watchful eye on each other, to ensure that every person does not infringe on each other's rights. He contrasts this scenario to what is already happening in several cities in the US and the United Kingdom, where, to reduce crime, cameras have been installed at every street corner, but are monitored constantly by a force of constabulary who are invisible to the public, and who could potentially abuse the power they are given. Brin's version, meanwhile, thrusts any body of authority out into the open, where they can be monitored to ensure their actions benefit the needs of the people [3].

By representing a positive outcome from what (he considers) the inevitable advancement of surveillance-related technologies, his viewpoint will hopefully liberate and encourage research towards designing systems that help, rather than hinder, protecting every person's civil rights.

5.4 User Modeling

Chapter 4 introduced the User Context Management layer of the Ki/o architecture, visible in Figure 4-1. Much of this layer is intended to be devoted to services that help make applications context-sensitive and personalized to the user. This section will describe possible approaches and services that this layer may eventually contain.

5.4.1 User Profiles

In order for an application to automatically provide personalized services to a user, a Ki/o kiosk must somehow identify the user, and obtain his or her preferences.[3]

User identification

Today, the first problem, user identification, can be solved a number of ways. The most common technique is to force the user to either provide a username and a password, or to present an RFid tag or smart card.

One of the research objectives of Ki/o for the near future is to incorporate an identification method that is transparent and unobtrusive so that it does not require any input from the user whatsoever. At the same time, this identification mechanism should make visible, and permit the user to modify, their *exposure level*, or how much identify information is being divulged and to whom.

The goal of unobtrusive identification is quickly becoming a reality as face recognition and other biometric identification techniques improve. Plans for a simple face recognizer and voiceprint identifier for the kiosk are currently being devised. For the latter goal, one scheme has so far been implemented. If a user is carrying a H21 handheld IE equipped with a Cricket indoor location system, the handheld can be made to provide the kiosk with relevant information about the user automatically

[3]Some research has surrounded early-customization of web pages through anonymous web access logs. See [30]for more information.

as the user approaches it. At the same time, since the handheld is under the user's control, it is left to users to directly set their exposure level.

Task Modeling

Any data that a system knows about a person could potentially be used by an adaptive user application to personalize and improve the user interface for the person. This data as a whole constitutes the user's *user profile*[4].

Determining what this data consists of, how it is captured, and how, precisely, applications could improve the user experience are currently active areas of research in the Intelligent User Interface (IUI) community. An example of such research include algorithms that attempt to extract and learn users' preferences based on observations about how a user manipulate an application's GUI. These algorithms may attempt to then deduce their task, and attempt to predict, automate or complete their task for them automatically [13].

The PLANLET plan-based representation for Metaglue is the Ki/o project's first step in developing tools for representing current user's task state and potential next task [27].

[4]also sometimes referred to as a *digital persona*

Chapter 6

Conclusion

Today, tiny microcontrollers are seemingly disappearing into nearly everything we touch, from kitchen toasters down to individual light switches. But even as people are becoming more reliant on computers for their day-to-day tasks, computers systems are continuing to become ever more complex. While computer technology has great potential to automate mundane tasks and to otherwise enhance people's lives, current trends in the design of applications and systems force everyday users to fight ever-changing, uphill battles to understand and effectively use each new revision of their computer-related technologies.

Ubiquitous computing research as first portrayed by Mark Weiser hopes to improve people's lives by allowing people to focus on what they want to accomplish, rather than on the technological tools themselves. One way to achieve this goal is to make computers the party to bridge the human-computer knowledge gap, by understanding users as people within their natural human habitats. Intelligent Environments form a first step in this direction, and research surrounding how to make these systems able to accurately sense, perceive, interact, and communicate with users is the focus of ubiquitous computing research today.

The Ki/o project is a response to studies indicating the role of random, informal encounters in spurring collaborations among the highly specialized knowledge workers of the modern workplace. Today, office layouts are choosing mobile, open arrangements that emphasize transitional spaces with the hopes that these will en-

courage random encounters that lead to informal collaborations. However, when random encounters do happen in these spaces, most of these spaces are not yet equipped with any means by which knowledge workers can access their valuable digital information or communications resources. Ki/o provides a way by which people can access these resources, thereby saving users from having to constantly carry mobile devices with them.

The work described in this paper represents an initial attempt at a complete design for an intelligent environment for such public spaces, encompassing both hardware and software components. Chapter 3 discussed a number of physical design considerations and provided glimpses at several simple physical installations. The agent-based software framework, described in chapter 4, provides a multitude of system services needed to build and run Ki/o applications. The software has already proven to be quite flexible and capable for building the initial prototype kiosk deployed in the laboratory. Unfortunately, a more formal evaluation of the framework is difficult at this early stage, because so few applications and kiosks have been completed. The future work chapter outlines the immediate future directions for this project and illustrates that, indeed, this investigation has only just begun. Hopefully, the rapid evolution of this project will yield exciting new developments and further research questions for several years to come.

Bibliography

[1] Jason Murray Bell. An A.P.I. for Location-Aware Computing. M.Eng. Thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2003.

[2] V. Bellotti and A. Sellen. Design for Privacy in Ubiquitous Computing Environments. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*, pages 77–92. Kluwer, 1993.

[3] David Brin. *The Transparent Society: Will Technology Force Us to Choose Between Privacy and Freedom?* Perseus Publishing, June 1999.

[4] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, pages 14–23, April 1986.

[5] Rodney Brooks. The intelligent room project, 1997.

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

[7] D. Chai and K. Ngan. Automatic face location for videophone images, 1996.

[8] Andrew D. Christian and Brian L. Avery. Speak out and annoy someone: experience with intelligent kiosks. In *Proceedings of the CHI 2000 conference on Human factors in computing systems*, pages 313–320. ACM Press, 2000.

[9] Michael Coen, Brenton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, and Peter Finin. Meeting the computational needs of intelligent environments: The metaglue system. In *Proceedings of MANSE'99*, Dublin, Ireland, 1999.

[10] Michael H. Coen. Building brains for rooms: Designing distributed software agents. In *AAAI/IAAI*, pages 971–977, 1997.

[11] Daniel D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, September 1991.

[12] James L. Crowley, Jolle Coutaz, and Franois Brard. Perceptual user interfaces: things that see. *Communications of the ACM*, 43(3):54–ff., 2000.

[13] Allen Cypher. Eager: Programming repetitive tasks by example. In *Proceedings of CHI'91*, pages 33–39, 1991.

[14] 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.

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

[16] E. H. Durfee and J. Rosenschein. Distributed problem solving and multiagent systems: Comparisons and examples. In M. Klein, editor, *Proceedings of the 13th International Workshop on DAI*, pages 94–104, Lake Quinalt, WA, USA, 1994.

[17] C.L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, pages 17–37, 1982.

[18] Krzysztof Gajos. Rascal - a resource manager for multi agent systems in smart spaces. In *Proceedings of CEEMAS 2001*, 2001.

[19] Jrg Geiler. Shuffle, throw or take it! working efficiently with an interactive wall. In *Proceedings of the conference on CHI 98 summary : human factors in computing systems*, pages 265–266. ACM Press, 1998.

[20] Nicholas Hanssens. A Framework for Speech Recognition in Intelligent Environments. M.Eng. Thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2003.

[21] Eric Horvitz. Uncertainty, action, and interaction: In pursuit of mixed-initiative computing. *Intelligent Systems*, pages 17–20, September 1999.

[22] Phillip R. Romig III and Ashok Samal. Devious: A distributed environment for computer vision. *Software - Practice and Experience*, 25(1):23–45, 1995.

[23] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *CHI*, pages 234–241, 1997.

[24] Ajay Kulkarni. A Reactive Behavioral System for the Intelligent Room. M.Eng. Thesis, MIT, Cambridge, MA, 2002.

[25] Ajay Kulkarni. Design Principles of a Reactive Behavioral System for the Intelligent Room. 2002. To appear.

[26] Raymond Kurtzweil. *Tribute to Michael Dertouzos*. kurtzweilAI.net, August 2001.

[27] Gary Look, Stefanie Chiou, Rob Kochman, Kevin Quigley, and Howard Shrobe. Spie: A self-adaptive, plan-based, intelligent environment. Technical report, 2003.

[28] A. Medl, I. Marsic, M. Andre, C. Kulikowski, and J. Flanagan. Multimodal man-machine interface for mission planning, 1998.

[29] D. Meyer. RFC 2365: Administratively scoped IP multicast, July 1998. See also BCP0023. Status: BEST CURRENT PRACTICE.

[30] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Improving the effectiveness of collaborative filtering on anonymous web usage data.

[31] Louis-Philippe Morency, Ali Rahimi, Neal Checka, and Trevor Darrell. Fast stereo-based head tracking for interactive environments. In *Proceedings of Conference on Automatic Face and Gesture Recognition*, 2002.

[32] Brad A. Myers. Using handhelds and PCs together. *Communications of the ACM*, 44(11):34–41, 2001.

[33] Jeremy Myerson. *The Creative Office*. Gingko Press, Corte Madera, California, 1999.

[34] Jakob Nielsen. Noncommand user interfaces. *Communications of the ACM*, 36(4):83–99, 1993.

[35] H. Penny Nii. Blackboard applications systems from a knowledge engineering perspective. *The AI Magazine*, 7(3), 1986.

[36] H. Penny Nii. The blackboard model of problem solving and the evolution of blackboard architectures. *The AI Magazine*, 7(2), 1986.

[37] George Orwell. *1984*. Secker and Warburg, London, 1949.

[38] Sharon Oviatt. Multimodal interfaces for dynamic interactive maps. In *Conference proceedings on Human factors in computing systems*, pages 95–102. ACM Press, 1996.

[39] Oxygen: Pervasive, human-centered computing. MIT Laboratory of Computer Science, Cambridge, Massachusetts, May 2002.

[40] Elin Rnby Pedersen, Kim McCall, Thomas P. Moran, and Frank G. Halasz. Tivoli: an electronic whiteboard for informal workgroup meetings. In *Proceedings of the conference on Human factors in computing systems*, pages 391–398. Addison-Wesley Longman Publishing Co., Inc., 1993.

[41] Brenton Phillips. Metaglue: A Programming Language for Multi-Agent Systems. M.Eng. Thesis, MIT, Cambridge, MA, 1999.

[42] Nissanka B. Priyantha, Allen K. L. Miu, Hari Balakrishnan, and Seth J. Teller. The cricket compass for context-aware mobile applications. In *Mobile Computing and Networking*, pages 1–14, 2001.

[43] L. Erman R. Reddy and R. Neely. The HEARSAY speech understanding system: An example of the recognition process. *EIII Transactions on Computers*, pages 427–431, 1976.

[44] Byron Reeves and Clifford Nass. Perceptual user interfaces: perceptual bandwidth. *Communications of the ACM*, 43(3):65–70, 2000.

[45] Jun Rekimoto. A multiple device approach for supporting whiteboard-based interactions. In *Conference proceedings on Human factors in computing systems*, pages 344–351. ACM Press/Addison-Wesley Publishing Co., 1998.

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

[47] Nitin Sawhney, Sean Wheeler, and Chris Schmandt. Aware community portals: Shared information appliances for transitional spaces. *Personal and Ubiquitous Computing*, 5(1):66–70, 2001.

[48] H. Schulzrinne, A. Rao, and R. Lanphier. RFC 2326: Real time streaming protocol (RTSP), April 1998. Status: PROPOSED STANDARD.

[49] John Scourias. Overview of the Global System for Mobile communications, October 1997.

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

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

[52] Ken Steele, Jason Waterman, and Eugene Weinstein. The oxygen h21 handheld. *ACM SIGARCH Computer Architecture News*, 30(3):3–4, 2002.

[53] 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.

[54] W. Teitelman and L. Masinter. The interlisp programming environment. *Computer*, 14(4):25–34, 1981.

[55] Matthew Turk and George Robertson. Perceptual user interfaces (introduction). *Communications of the ACM*, 43(3):32–34, 2000.

[56] Alex Waibel, Minh Tue Vo, Paul Duchnowski, and Stefan Manke. Multimodal interfaces. *Artificial Intelligence Review*, 10(3-4):299–319, 1996.

[57] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995.

[58] Nimrod Warshawsky. Extending the Metaglue Multi-Agent System. M.Eng. Thesis, MIT, Cambridge, MA, 1999.

[59] Mark Weiser and John Seely Brown. The coming age of calm technology. Technical report, Xerox PARC, 1996.

[60] Catherine G. Wolf and James R. Rhyne. Gesturing with shared drawing tools. In *INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, pages 137–138. ACM Press, 1993.