Building Agent-Based Intelligent Workspaces

Nicholas Hanssens, Ajay Kulkarni, Rattapoom Tuchida, and Tyler Horton
Artificial Intelligence Lab
Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*— *The idea of weaving technology into the backdrop of natural human interactions is a long-standing vision [1]. To realize that dream, a number of technologies need to be developed. Distributed software systems need to allow large numbers of software components to locate and communicate with one another. Interfaces between humans and computers need to enable natural, unencumbered interaction. Environments need to be aware of what users are trying to do so that they can offer appropriate assistance. Intelligent tools and applications need to be built on top of these components.*

The Intelligent Room project has taken an agent-based approach to building and organizing these components. In this paper, we present concepts, technologies, and applications that we have developed to address these needs, ranging from low-level communication infrastructure to distributed applications with multi-modal interfaces. In doing so, we situate these technologies in the context of Intelligent Environments (IEs) which enhance day-to-day activities in a business environment.

***Index Terms*—agents, intelligent environments, human-centric computing**

I. Introduction

Computing technology is "weaving" itself into the fabric of everyday life [1], but this emerging "Pervasive Computing" can become either a blessing or a curse. Traditionally, human computer interfaces have drawn people into the world of the desktop. We believe that for pervasive computing to be successful, we must build interfaces that work the other way around, drawing the computer into our natural world of human discourse; computers will need to communicate using speech and vision just as humans do. MIT's Project Oxygen [2] was started with these goals. One particular form that such human-centered technology can take is that of an Intelligent Environment (IE), a physical space that is perceptually enabled, that is capable of natural human interactions, and that provides both proactive and reactive services to a community of users. The Intelligent Room project is the component of MIT's Project Oxygen that seeks to construct such IEs.

An Intelligent Environment needs to control a wide range of physical devices: lights, audio/video equipment, telephone equipment, handheld computers, etc. Additionally, an IE needs to control a significant number of software components: messaging systems, personal file databases, schedule-keeping agents, and so on. A control system for an IE needs to provide a standard mechanism that manages components, enables communication between them, facilitates interactions between the user and the environment, and protects security and privacy.

Dynamic and mobile environments place additional demands on such a system. Mobile components need a mechanism for discovering and effectively using their surroundings, while stable environments need to be able to incorporate those mobile devices. Lastly, these control and coordination mechanisms need to be extensible to work over different physical spaces.

In this paper, we present a fictional scenario which illustrates our vision of how information technology can be woven into the background. Following the scenario we discuss our previous work, drawing out a set of driving principles for our research. We then organize our research into a multi-layered infrastructure capable of supporting such a scenario. Lastly, we present several applications, demonstrating how our infrastructure can be used.

II. The Intelligent Meeting

It's 9:45 am. You have a meeting in the main conference room in 15 minutes, but it looks as if you will still be stuck in your car on the freeway.

Already, Alice, Bob, and Carol are getting ready for the 10am marketing meeting to discuss the new product, iBoggle; you want to let them know where you are.

You dial the main conference room phone number, and ask the computer that answers to forward your call to whomever is running the morning meeting.

Alice answers. "Alice, I'm stuck on the freeway. Can we push back the meeting to 10:30?"

"Sorry, David, but Bob has another meeting at 11. Why don't we just meet remotely at 10?"

You agree and make your way over to the shoulder lane. At 10 o'clock, your cell phone rings. You flip it open, revealing a handheld computer that features a display, video camera, microphone, and a wireless connection to the car navigation display and audio system.

"Hello, David, we're ready to start."

To the computer you request, "*Computer, connect to the meeting*," and the computer responds with a chirp. A window displaying the live video feed from the room is opened on your handheld display, while another with the agenda is opened on the navigation display mounted in your car. Finally, as you are aware, your face has now been projected onto one of the room's walls.

"Alright, then," Alice asserts, "let's get this going. *Computer - start the meeting*." Back in the conference room, the meeting agenda is projected onto a wall, and the first agenda item is highlighted: "David's Presentation on 'Adapting Traditional Games in Intelligent Environments: iBoggle."'

As you load your slides, they are placed on the main wall and each attendee's laptop. Simultaneously, you notice through the video feed, the lights are dimmed.

"When Parker Brothers® created Boggle® 25 years ago, they probably did not imagine the influence it would have on generations of children and adults. As we see here in an old advertisement..." As you launch into your presentation, you use your pen to draw attention to specific points, and accordingly your notes appear on each meeting attendee's laptop.

A few slides later, Carol leans towards the projection of your face and raises her hand with a question. An icon pops up in the corner of your display, and after finishing your thought you say, “Yes, Carol.”

“What other games has Parker Brothers developed? Also, do we need to get their permission to use their trademark?”

“That’s a good question. Let’s check their website. *Computer – search the Internet for ‘Parker Brothers’.*” Your computer beeps and displays the results, organized by various categories, in a window on the navigation display. Using your pen, you prune through the categories to find the official Parker Brothers website: “*Delete all results except those in a ‘Games’ or ‘Entertainment’ category.*” Your computer complies and beeps. *“Recategorize by types of Entertainment.”*

You then look at the results in the appropriate branch of data and find a list of games that Parker Brothers has produced. “*Computer – display this page on the main display in the conference room.*”

“As you can see, Carol, Boggle is one of the more popular Parker Brother games. But I think our development team put together ‘iBoggle’ without much regard to the trademark. After this meeting, I’ll check with our legal department and get back to you.”

You continue through your slides, describing the layout of the Boggle board.

“David, may I modify the layout to make a suggestion?,” Bob interjects. After you grant him permission to load the layout editor, he continues: “Is there any way we can enlarge the logo and move it over here?” As he makes his point, he gets up, picks up the pen, and drags the logo in your Boggle layout across the main display.

“We can’t move it there, Bob, because it gets in the way of the score sheet. But how about a little higher and to the left?” On your handheld, you use your pen to drag the logo diagonally from where Bob placed it. “And I’ll see if we can enlarge it.”

At this point, your computer beeps to remind you that you only have two minutes left in the presentation. You skip ahead to your last slide and summarize your major points.

“Thank you, David, for the presentation,” Alice remarks. “*Computer, move on to the next agenda item.*” On the agenda display, which is back to the front of the navigation panel, “David’s Presentation” is checked off. The next agenda item, “Set up the New Product Focus Group,” is now highlighted.

While we are still some time away from realizing the interactions in this short fictional tale, our current research is bringing us closer. As this paper describes our research, it focuses on,

- Transparent, context-based reactions to user behavior
- Automatic resource allocation, allowing users and applications to share resources
- Stable, natural cross-space collaboration

In the next section, we will discuss the primary principles driving our research. Following that section, we describe our research and argue how these above goals are addressed.

III. Driving Principles

In building our framework for IE’s, we have developed several guiding themes and design techniques which we consider central to our vision of how an IE should operate. This section lays out these themes and defines their terms.

First, we discuss the overall *design philosophy* of how we organize an IE, from low-level communication infrastructure to high-level applications. Then we develop the idea of personal and physical *spaces* into which agents are organized. Finally, we introduce the idea of *context* as an important mechanism for interacting with the user appropriately.

A. Designing an Intelligent Space

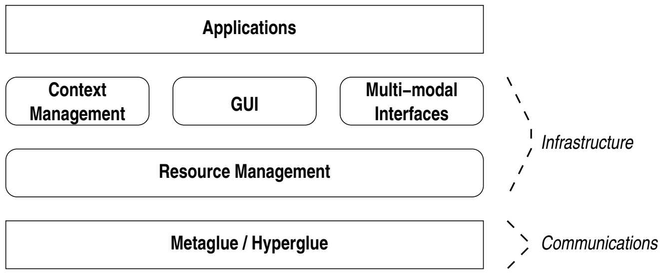

Fig. 1. Intelligent Room Design

The Intelligent Room project seeks to enable a variety of tools and applications that can be used flexibly and naturally. Starting with the guiding principles outlined by Coen [3], we have created several component layers at various levels of abstraction (Figure 1).

At the lowest level, the Room has a communications layer which consists of individual software *agents*. For our purposes, we define an agent to be any software object with the ability to communicate by exposing functionality to other agents running within the network. This definition deliberately under-specifies the capabilities of an agent. As such, an agent can be anything from a software control for a light switch with no user interface to a full-featured word processing application. Agents, hence, form the underlying communication structure of the Intelligent Room. Our implementation of this layer is discussed in section IV.

Using basic agents, we have constructed a *resource management* layer of organization. In addition to an arbitration mechanism, this layer provides a high-level resource discovery service. For example, using just the communication layer, an agent would need to know the exact name of a specific light controller to turn on a light. Using the resource-management layer an agent can simply ask, in general, for a light to be turned on in a particular space. Similarly, this layer also addresses access control as a high-level problem. This layer is critical to system flexibility, particularly when agents need to move from one environment to another, and is discussed further in section VII.

On top of resource management structures, we have developed three parallel layers: *context management*, *graphical user interfaces*, and *multi-modal user interfaces*.

The context management layer contains tools to observe and use contextual information. Context is discussed in more detail in section III.C and our context management infrastructure is presented in section VI.

The graphical user interface (GUI) layer contains infrastructure to control graphical interfaces to applications. In particular, this layer is responsible for allowing applications to adapt to available display resources. This technique is discussed further in the section IIX.

Alongside GUI's, the multi-modal interface layer contains methods for other modes of user input, such as gesture and speech recognition. Speech, is particular, is an important mechanism for interacting with an IE. This topic is developed further in section V.

At the top, we find *applications*–parts of the Room which interact directly with users. Because they are built on top of the Room's infrastructure base, our applications are very distinct from their conventional counterparts, such as desktop spreadsheet packages. Our infrastructure allows developers to build flexible, dynamic applications which can adapt effectively to different situations and environments. We describe several existing examples in section IX.

B. People and Spaces

Within intelligent spaces, there are three types of interactions: interactions between a space and users, between multiple spaces, and between multiple users. The first occurs when one or more users are in the room. Users can request a service from the room, or the room can react to users based on their current activity or other contextual information. Interactions between spaces occur when one space needs to request services from another space, such as a server room sending a request to a control room to lower the temperature. Interactions between users can be as local as talking face to face, or as remote as in chatting via personal digital assistants(PDAs). Finally, different combinations of these three types can occur. For example, two people communicating across multiple spaces involve interactions between a user and a space and between two spaces.

Regardless of interaction type, an IE needs to arbitrate requests for access to resources between different entities. Such constraints make it necessary to partition devices and agents into groups that manage their own resources and enforce their own access control [4]. In order to address these needs, we introduce the concept of a *society* – a group of agents that act on behalf of a user or a space. For example, devices in the meeting room, such as projectors, would be controlled by agents in the Meeting Room's society. On the other hand, a Reminder agent running on Alice's PDA would belong to Alice's society.

Each society uses an *Ambassador* agent as a proxy to initiate communication with other societies. Routing communication through an Ambassador agent allows a society to selectively expose its resources and utilities to other societies.

We describe how a space can vary its reactions to users via contextual information below and in section VI. The topic of resource allocation, including resource management and access control, is discussed in section VII.

C. Context

Underlying the Intelligent Room is the fundamental ability of the Room to respond to a user's needs. In order to accomplish this, the Room collects *contextual information* to attempt to discern what the user is currently engaged in and what she is trying to do.

Previous work in the area of context management in an IE defines context as information about the "situation of an entity," where an entity is a "person, place, or physical or computational object" [5]. From this information, which usually includes the location [6], number, and current state of persons and objects in the IE, an application would be able to infer current user needs and provide better services to the user [7]. Such an application, i.e., one that uses this contextual information, is said to be *context-aware*.

Context management in HCI applications is a challenging problem [5]. The complexity of maintaining contextual information is two-fold: first, in collecting the right information, and second, in knowing what to do with the information once you have it.

For collecting the right information, an IE can provide a unique set of tools. Data collected from basic devices in an IE (cameras, microphones, sensors, etc.) can be used for deriving a sense of the context of the environment.

In addition, after collecting the right information, an IE presents methods to effectively incorporate context-based knowledge to improve huyman-computer interactions. While the bulk of these methods are presented in section VI, context management is an underlying theme to much of infrastructure presented throughout the paper.

IV. Communication

At the lowest level of an IE, agents need to speak to one another to control and coordinate tasks. Additionally, agents need to rely on a stable backbone that ensures that other components remain alive and functional. In order to meet these needs, we have developed *Metaglue* and *Hyperglue* – two projects that are aimed at providing these services. Metaglue provides agents with simple, robust, fault tolerant communication mechanisms. Hyperglue then extends Metaglue's services to a wider area by allowing agents to locate and communicate with other societies.

A. Metaglue

An IE is composed of a dynamic set of hardware and software components. New devices or software modules can be added to the system, old ones can break down, computers can crash, and mobile devices can enter and leave frequently. In order to keep technology in the background, an IE needs infrastructure to automatically handle these circumstances.

To that end, devices need to be able to reliably and robustly communicate with one another in order to coordinate their actions. Metaglue [8], [9] is a system that provides IE components with low-level support satisfying these needs. Such components can range from low-level controllers for devices, such as projectors, to high-level agents, such as meeting management applications. Metaglue features two forms of communication, and a fail-over mechanism for restoring agents after crashes.

Agents can communicate by requesting a *stub* for another agent, and calling methods directly on that agent through the stub. This process can be entirely mediated by the resource allocation system, which is discussed more in Section VII. The stub also carries a fail-over mechanism: if an agent crashes, the stub automatically restarts it, possibly on another computer, and reestablishes the connection [10]. Restarted agents can then automatically recover their state. Agents may further request to be automatically restarted.

Agents can also communicate using a publish-subscribe mechanism, whereby an agent can listen for specific classes of messages and broadcast messages using those same classes. This allows agents to communicate information without needing to know which agents it is speaking to.

Metaglue provides a basic level of support for agents that allows developers to abstract away concerns of communication and fault tolerance. These are critical functions that need to be implemented at the lowest level of an IE, as higher-level agents all depend on these elements to maintain a stable environment.

B. Hyperglue

Hyperglue is a communication layer component that enables wide scale communications between societies. As mentioned when discussing interactions between users and spaces (see section III.B), each person and space has its own society of agents and an Ambassador agent which represents them. These Ambassador agents need to locate each other in order to request resources and exchange information. As a result, a society discovery mechanism is necessary for inter-society communication.

To meet this requirement, Hyperglue uses the Intentional Naming System (INS) [11] as a society discovery mechanism. This allows an Ambassador agent to advertise its society and to locate other societies. Interactions in the scenario rely on such a mechanism. For example, when David asks his computer to "connect to the meeting," his computer does not necessarily know where to contact the meeting room's society. Agents in David's society use Hyperglue to locate the Ambassador agent for the room's society and send it a request to show his slides using projectors in the room.

Hyperglue allows societies to communicate with each other on a wide scale, enabling applications to work across multiple spaces.

V. Speech

Speech recognition is an important tool for developing natural interactions with IEs. Towards that end, we have experimented with a wide range of speech technologies and developed a substantial speech infrastructure [12].

A. Speech Infrastructure

Several agents comprise our current speech infrastructure. At the highest level, the Grammar Center agent exposes speech functionality to other agents in the system. An individual agent can register a grammar of phrases that it recognizes with the Grammar Center. In order to make use of existing speech technologies, all grammars are currently written in the Java Speech Grammar Format (JSGF).

When a user speaks to the Room, the low-level recognizer checks the utterance against all active grammars and propagates any matching phrases back to the agent which registered the grammar containing the match. That agent then takes the phrases and performs the appropriate action.

This speech system allows individual agents and applications to enable simultaneous, dynamically-controlled speech input without having to directly interact with the underlying recognition technology.

B. Speech and Context

Context plays an important role in speech recognition. For example, when a map application zooms in to a specific country, the names of the visible regions or cities become pertinent. Individual applications can use their internal context, then, to dynamically activate, change, and deactivate grammars [12].

Another of our projects aims to make a heavier use of context within speech recognition. In particular, we intend to add activity context to the semantic specification of agents. For example, in the scenario, David runs FIRE, an information retrieval engine described in section IX.A, to do a search. This activity context would trigger certain phrases, such as "display this page somewhere else" which, while not being one of FIRE's commands, is an important function in that context. This higher level integration of grammars into context management will also help to dramatically increase recognition rates by decreasing the number of grammars that need to be active at any given time. Activity context is discussed further in the next section.

VI. Context Based Reactive Behaviors

As described earlier in section III.C, a context-aware Intelligent Room will be able to provide more and better assistance to its users. In this section, we focus on constructing a representation for context and using that representation to understand user needs and expectations.

Before the system can create this representation, it needs to collect data describing user actions and behavior. We rely on various components to detect this information: computer vision for head, arm, and person tracking [13], [14]; sensors for motion, location, pressure, and temperature sensing; and enhanced device controllers for device state detection. In short, we collect information about the current state of both the Room and the occupants within.

The following three subsections describe *ReBa*, a reactive behavioral system that not only facilitates building a context representation, but also uses that knowledge to provide relevant services to the user. First, ReBa's basic representation of activity-centric context is explained. Next, how these basic building blocks are combined into more complex representations is described. Finally, the problem of conflicting behaviors arising from this architecture is identified, and a solution is proposed.

A. Basic Behavior Representation

For the room to correctly react to a user's action, it needs to situate that action within its context. Knowing what task the user is currently performing can help do so. We designed ReBa with this notion in mind. That is, by observing user actions and following pre-specified definitions of tasks, ReBa recognizes the ongoing task(s) within the Room and presents an organized method to structure Room reactions.

ReBa observes user actions by collecting data from other agents. Given low level contextual information (e.g., where a user is located, which device was just turned on), ReBa builds a higher level representation of the current task, or activity, performed in the Room by the current users. We call this representation *activity context*– information about the current ongoing task, or activity, performed in the Room by the current users.

Other research also argues in favor of activity-based context modeling. Prekop and Burnett argue that an activity-centric model can be used to develop context-aware applications [7]. According to Canny and Fisher, activity-centric computing helps decompose user actions into their significant components: "Activity theory divides human behavior into a hierarchy."[15]. While Prekop and Burnett focus on knowledge-management, and Canny and Fisher on understanding relationships between people and data, ReBa focuses on using this contextual information to reach goals more specific to an IE. A detailed list of design principles can be found in [16].

Each activity (or sub-activity) that a user performs is represented by a software agent, called a *Behavior agent*. For example, we have a Movie behavior, a Meeting behavior, a Casual behavior. Some behaviors are subsets of others: we also have a Presentation behavior and a Brainstorming behavior, both particular types of a Meeting.

At the lowest level, a behavior contains a rule that, for a given user action (input), performs a reaction (output). For example, for user entry into the Room, one rule may instruct ReBa to turn the lights on. Rules are then organized into groups called *actions*. While no restrictions on grouping rules exist, for understandability and ease of maintenance rules are often grouped if they affect similar devices or share functionality. A set of rules affecting the lights, for example, would be organized in a "lights" action, while another set greeting the user into the Room would be organized into a "greeting" action. A behavior is thus a collection of actions.

B. *Layering Behavior Building Blocks*

Based on the order of user actions, the system activates a series of behaviors. If an activity is being performed within another activity, the behavior corresponding to the one started second is activated on top of the first. In the scenario, David starts a presentation within a meeting; the Presentation behavior is activated on top of the Meeting behavior.

This practice of layering behaviors helps to mirror user expectations and prevent redundancy. When David loads his slides, he expects the system to recognize that he is giving a presentation within the context of a meeting. If he had given a sideshow within the context of a casual gathering (e.g., showing images of his last Las Vegas vacation), David would expect the Room to react somewhat differently: he would expect reactions specific to a presentation (e.g., dim the lights) but not a meeting (e.g., don't display the slides on an attendee's laptop).

C. *Conflict Resolution*

Layering behaviors on top of each other introduces the problem of conflicting behaviors. If, say, one behavior wants to turn the lights on for user entry, while another wants to keep them off, the system needs to resolve this conflict.

Conflict resolution is a central problem in designing Automatic Behavior systems [17]. In ReBa, this problem is addressed through the arranging of behavior rules into actions: overriding at the actions level allows for a simpler method of resolution than that at the rules level. In the earlier conflict example, one behavior's lights action can override that of the other. Generally, when a behavior is activated, it overrides any action (within the active behaviors) that shares a name with one of its own actions.

Despite the system's best intentions and efforts to mirror user expectations, it may every now and then not completely satisfy user needs. With this limitation in mind, future work in ReBa will collect user feedback and adapt its actions accordingly.

VII. Resource Allocation

An IE presents a common set of resources (displays, speakers, lights, and devices) to all agents, both software and human, within the environment. For these agents to share these resources, some mechanism to control who can use which resources is necessary.

Also, as our infrasctructure is deployed in a number of different spaces, layers of abstraction between the low level devices and the higher level services become more important.

In this section, we present two components that allocate resources among software and human agents. The first, *Rascal*, allocates services among software agents. The second enforces *access control* in the Intelligent Room.

A. *Rascal*

In an agent-based IE like the Intelligent Room, coordinating agents is often difficult due to physical constraints. Allocating a resource that has the capability to handle only a limited number of requests is difficult. When a resource's limits have been reached, an arbitrator needs to decide whether the extra requests need to be sent to another resource providing a similar service, or whether old requests should be dropped by the resource in favor of new requests. Also, different IE's (e.g., a living room, an office, a car) perform different functions, and are thus equipped with different kinds and quantities of devices. A layer of abstraction is necessary to ensure that applications can be created independent of the physical characteristics of any given space [18].

Rascal [19], a system for managing resources in the Intelligent Room, is designed to fill this role by establishing a level of indirection between applications and physical and software resources provided by a space. By creating this separation, Rascal allows applications in the Intelligent Room to request services rather than specific devices. Its knowledge base contains information about resource requirements; its constraint satisfaction engine arbitrates requests for similar resources; its general framework provides communication with other agents.

Through these three components, Rascal allows many applications to coexist in a variety of environments. For example, within the scenario Rascal presents the agenda and the live video feed on two different displays (the handheld display and the navigation panel). When David starts searching for the Parker Brothers website, the results are shown on the navigation display on top of the agenda. Neither David nor an application in use by David needs to specify which data should go on what display. And if David were in another space with completely different displays, e.g., the conference room, Rascal would still know what to do.

B. Access Control

In the scenario, Bob can modify the layout of the Boggle board only after David grants him a permission to do so. This example demonstrates the role of access control in the IE.

Access control mechanisms (ACMs) restrict resources from unauthorized users. The most widely used mechanism is an Access Control List (ACLs), which maps each user to a list of devices or resources that they have permissions to access. While ACLs have been successfully used in traditional computer systems such as MS Windows® and Linux, there are subtle problems that arise when ACLs are deployed within IEs. For example, a user outside a meeting room should not be able to control lighting in the room during a meeting. Since an ACL does not use any contextual information in its mechanism, implementing the above example using ACL would be difficult.

Access control in IEs should allow contextual information to be incorporated into rules [20]. In the above example, the pertinent contextual information consists of the location of the user and the mode of the room.

Currently, we are implementing an ACM that integrates contextual information using [20] and [21] as guidelines.

VIII. GUI Management

In an IE, multiple users often need to interact with the same data and the same applications from different locations and interfaces. Similarly, users need to access their personal applications regardless of their location and available technology.

In the scenario, David needs to access his slides from his car and to simultaneously broadcast them on the projector in the meeting room. Additionally, his PDA, having a limited display area, needs to manage the display of David's slides, the video feed from the meeting room, and the iBoggle design layout application. Lastly, he needs to share the layout manager with Bob, who is back in the meeting room.

To support these needs, we created a distinction between an application agent and the graphical user interface (GUI) of that application. To manage the user interfaces transparently, we created the *GUI Manager agent*, a component which automates dispatch and control of user interfaces. A GUI Manager agent runs locally on each display and can request a GUI from an application to load it locally. The basic principle is that an application agent or set of agents can run anywhere while the interface is developed independently (or possibly automatically) to suit the specific situation.

Because the GUI Manager remains independent of the application itself, different user interfaces to the same application can be easily deployed without the need to make changes to the application. This functionality allows user interfaces to adapt dynamically to local resources, such as David's car; to share applications across users, such as the iBoggle design application; and to allow users to keep in contact with their principal tools, such as David's access to his slides, regardless of location and surroundings.

IX. Intelligent Room Applications

This section discusses several intelligent room applications and situates them in the context of the infrastructure they use. In doing so, we demonstrate that our infrastructure is capable of supporting high-level applications in an IE.

A. FIRE

FIRE, an Information Retrieval Interface for Intelligent Environments [22], is designed to help users find data on the Internet. It specifically takes advantage of the multi-modal nature of the Intelligent Room. David uses FIRE when he searches for the Parker Brothers website and prunes through the results on his navigation display. The scenario illustrates how FIRE would make use of three infrastructure components: speech, the GUI Manager, and Rascal.

The Intelligent Room allows FIRE users to rely on modes of communication that are more natural than the traditional mouse-keyboard model. A user can utilize speech and gesture (through a *pen-mouse*, a pointing device in place of the mouse) to convey commands and requests to the system. Gestures with the pen-mouse, while analogous to those with a mouse, can seem more natural on a display or on the large walls of a conference room.

Using the GUI Manager, FIRE organizes large groups of data into categorical trees onto multiple displays. The user can then recategorize these trees according to his own preferences. In showing this data, FIRE uses Rascal to obtain displays. In David's car, FIRE is only allowed to display on the navigation panel, as the other, namely David's handheld device is occupied with the live feed from the Room. Rascal understands that neither displaying the agenda nor granting FIRE more than one display is more important than showing the live feed.

As a natural and effective method of searching and browsing documents on the World Wide Web, FIRE is an example of a business application within an IE that uses the infrastructure of the IE.

B. Meeting Manager

The Meeting Manager (MM) [23] is an application aimed at organizing information about a meeting by recording major events and arranging those events into a data structure that supports complex queries [24].

When Alice says "Computer - start the meeting," the MM uses speech recognition and contextual information to deduce what meeting Alice is referring to so it can query its database and load the appropriate agenda, including unfinished items from the last meeting. The MM then notifies ReBa of a context change, which causes the room to react by starting a meeting. Meanwhile, Rascal checks available projectors and selects those with the appropriate size and resolution for the current tasks.

C. iBoggle

The iBoggle Application (iBA) is based on the popular Parker Brothers game of the same name. This application demonstrates that various infrastructure components can function properly when communicating across societies.

The iBA employs the GUI Manager, thus enabling the user's game board to appear on a variety of displays including computer monitors, projection screens, and handheld displays. Coupling this versatility with the Rascal allows the iBA interface to appear on whichever display is most appropriate, based on the state of all other displays in the IE.

Additionally, iBA accepts multi-modal input from speech, keyboards, pens, and accelerometers. The cross-society communication of these multi-modal inputs allows the iBA to function in across variety of IEs.

X. Contributions

Weiser presented a vision of moving computers off the desktop and into the background as transparent tools that a user can apply to everyday situations [1]. The Intelligent Room project is dedicated to reaching that goal.

Our new component technologies for IEs are important steps in this direction. Hyperglue enables wide-scale communication across different spaces while the GUI Manager tool facilitates application sharing and interface distribution across those spaces. ReBa, additionally, introduces context-driven behaviors to our IE. Our previous research and these tools, when combined, present a general set of design principles and organizational techniques for building effective intelligent environments.

Metaglue and a number of components and applications described above have been deployed in numerous offices and conference rooms around the Artificial Intelligence Laboratory at MIT as well as at the Nokia Research Center in Burlington, MA and at the Information Technology Division of the Australian Department of Defense.

XI. Acknowledgements

We would like to thank Krzysztof Gajos, Kimberle Koile, Alice Oh, Stephen Peters, and Howard Shrobe for their advice and input on this paper.

XII. Funding

This work was funded by Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Phillips Research under the MIT Project Oxygen partnership and by DARPA under contracts F33615-00-C-1702, N66001-99-2-891702, and N66001-00-C-8078, administered by AFRL/IFSC, the Office of Naval Research, and SPAWAR, respectively.

References

[1] Mark Weiser, "Computer of the 21st Century," *Scientific American*, vol. 265, no. 3, pp. 94–104, September 1991.

[2] M. Dertouzos, "The Oxygen Project," *Scientific American*, vol. 282, no. 3, pp. 52–63, August 1999.

[3] Michael Coen, "Design Principles for Intelligent Environments," in *Proceedings of AAAI'98*, 1998.

[4] Krzysztof Gajos and Howard Shrobe, "Delegation, Arbitration and High-Level Service Discovery As Key Elements of a Software Infrastructure for Pervasive Computing," 2002, In submission.

[5] Anind K. Dey, Gregory D. Abowd, and Daniel Salber, "A Context-Based Infrastructure for Smart Environments," in *Managing Interactions in Smart Environments*, Paddy Nixon, Gerard Lacey, and Simon Dobson, Eds., Dublin, Ireland, December 1999, MANSE.

[6] Justin Lin, Robert Laddaga, and Hirohisa Naito, "Personal Location Agent for Communicating Entities (PLACE)," in *Proceedings of Mobile CHI*, 2002.

[7] Paul Prekop and Mark Burnett, "Using Activity-Centric Context to Support Ubiquitous Computing," 2002, Information Technology Division, Department of Defence, Australia. In submission.

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

[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 *Managing Interactions in Smart Environments*, Paddy Nixon, Gerard Lacey, and Simon Dobson, Eds., Dublin, Ireland, December 1999, MANSE.

[10] Nimrod Warshawsky, "Extending the Metaglue Multi Agent System," M. Eng Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

[11] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, "The Design and Implementation of an Intentional Naming System," in *Proc. 17th SOSP*, 1999, pp. 186–201.

[12] Michael Coen, Luke Weisman, Kavita Thomas, and Marion Groh, "A Context Sensitive Natural Language Modality for an Intelligent Room," in *Managing Interactions in Smart Environments*, Paddy Nixon, Gerard Lacey, and Simon Dobson, Eds., Dublin, Ireland, December 1999, MANSE.

[13] Louis-Philippe Morency, Ali Rahimi, Neal Checka, and Trevor Darrell, "Fast Stereo-Based Head Tracking for Interactive Environment," in *Proceedings of the Int. Conference on Automatic Face and Gesture Recognition*, 2002, To appear.

[14] Trevor Darrell, David Demirdjian, Neal Checka, and Pedro Felzenswalb, "Plan-view Trajectory Estimation with Dense Stereo Background Models," in *Proceedings of the International Conference on Computer Vision*, 2001.

[15] John Canny and Danyel Fisher, "Activity-Based Computing," in *Proceedings of CHI*, The Hague, The Netherlands, 2000.

[16] Ajay Kulkarni, "Design Principles of a Reactive Behavioral System for the Intelligent Room," in *Bitstream: The MIT Journal of EECS Student Research*, 2002, To appear.

[17] Steven A. N. Shafer, Barry Brumitt, and JJ Cadiz, "Interaction Issues in Context-Aware Intelligent Environments," *Human-Computer Interaction*, vol. 16, 2001.

[18] Krzysztof Gajos, Luke Weisman, and Howard Shrobe, "Design Principles for Resource Management Systems for Intelligent Spaces," in *Proceedings of The Second International Workshop on Self-Adaptive Software*, Budapest, Hungary, 2001, To appear.

[19] Krzysztof Gajos, "Rascal - a Resource Manager for Multi Agent Systems in Smart spaces," in *Proceedings of CEEMAS 2001*, 2001.

[20] Rattapoom Tuchinda, "Access Control Mechanism for Intelligent Environments," in *Bitstream: The MIT Journal of EECS Student Research*, 2002, To appear.

[21] Sofia K. Tzelepi, Dimitrios K. Koukopoulos, and George Pangalos, "A Flexible Content and Context-based Access Control Model for Multimedia Medical Image Database Systems," in *Proceeding of ACM MMSig'01*, Ottowa, Canada, 2001.

[22] Krzysztof Gajos and Ajay Kulkarni, "FIRE: An Information Retrieval Interface for Intelligent Environments," in *Proceedings of International Workshop on Information Presentation and Natural Multimodal Dialogue IPNMD*, Verona, Italy, December 2001.

[23] Alice Oh, Rattapoom Tuchinda, and Lin Wu, "Meeting Manager: A Collaborative Tool in the Intelligent room," in *Proceedings of Student Oxygen Workshop*, 2001.

[24] Stephen Peters, "Using Semantic Networks for Knowledge Representation in an Intelligent Environment," 2002, UBICOMP 2002, In submission.