The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms

The interactive workspaces project explores new possibilities for people working together in technology-rich spaces. The project focuses on augmenting a dedicated meeting space with large displays, wireless or multimodal devices, and seamless mobile appliance integration.

Brad Johanson, Armando Fox, and Terry Winograd
Stanford University

The interactive workspaces project started at Stanford University in 1999 to investigate human interaction with large high-resolution displays. The project initially operated in a busy lab in which the display proved to be no more than a curiosity since it could not be used for long periods of time and offered little integration with other devices. It became clear that the potential of a large display device would emerge only by embedding it in a ubiquitous computing environment that could sustain realistic interactive use. The interactive workspaces project therefore began to design and use rooms containing one or more large displays that had the ability to integrate portable devices.

The idea of ubiquitous computing[1] encompasses many different kinds of settings and devices. We chose to narrow our focus by

- Investigating how to map a single defined physical location to an underlying systems infrastructure and a corresponding model of interaction[2]
- Emphasizing the use of large interactive walk-up displays, some using touch interaction
- Collaborating with other research groups, both within and outside the field of computer science, to construct nontoy applications

We constructed several versions of our prototype interactive workspace, which we call the iRoom, created a software infrastructure for this environment, called iROS, and conducted experiments in human-computer interaction in the workspace. We also assisted outside groups in using our technology to construct application suites that address problems in their own domains, and we deployed our software in production environments.

Overview, goals, and contributions

As we began to construct the iRoom, we developed some guiding principles:

- *Practice what we preach.* From the beginning we used the iRoom as our main project meeting room and employed the software tools that we constructed. Much of our continuing research has been motivated by our frustration at encountering something we could not accomplish in the iRoom.
- *Emphasize colocation.* There is a long history of research on computer-supported cooperative work

1536-1268/02/$17.00 © 2002 IEEE

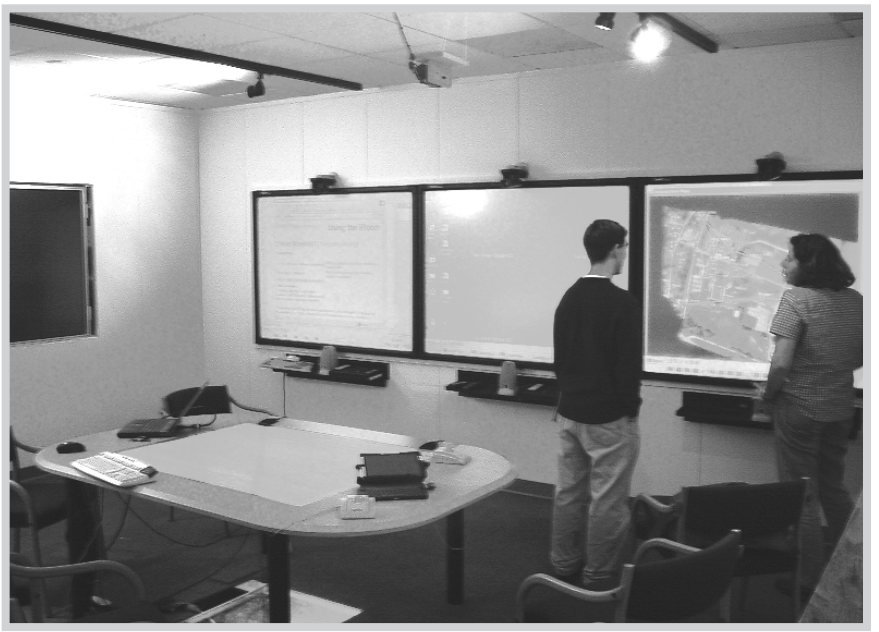


Figure 1. A view of the interactive room (iRoom).

for distributed access (teleconferencing support). To complement this work, we chose to explore new kinds of support for team meetings in single spaces, taking advantage of the shared physical space for orientation and interaction.

- *Rely on social conventions.* Many projects have attempted to make an interactive workspace smart.[3,4] Rather than have the room react to users, we chose to focus on letting users adjust the environment as they proceed with their task. In other words, we set our *semantic Rubicon*[2] so that users and social conventions take responsibility for actions and the system infrastructure is responsible for providing a fluid means of executing those actions.
- *Aim for wide applicability.* Rather than investigating systems and applications just in our specific space, we decided to investigate software techniques that would also apply to differently configured workspaces. We wanted to create standard abstractions and application design methodologies that apply to any interactive workspace.
- *Keep it simple.* At both the interface and software development levels, we tried to keep things simple. On the human-interface side, we faced a fundamental tradeoff in interaction design between the necessity of supporting diverse hardware and software and the need to provide an interface simple enough so that people would use it. On the software development side, we tried to keep APIs as simple as possible, both to make the client-side libraries easier to port and to minimize the barrier-to-entry for application developers.

The iRoom is our second-generation prototype interactive workspace (see Figure 1). Several other iRooms have been created at Stanford and elsewhere (see the sidebar "The iRoom and Beyond: Evolution and Use of Deployed Environments"). The iRoom contains three touch sensitive white-board displays along the side wall and a custom-built 9 megapixel, 6-foot diagonal display called the interactive mural. In addition, there is a table with a 3×4 foot display that was designed to look like a standard conference-room table. The room also has cameras, microphones, wireless LAN support, and several wireless buttons and other interaction devices.

We started our research by determining the types of activities users would carry out in an interactive workspace. Through our own use, and through consultation with collaborating research groups, we arrived at the three following task characteristics:

1. *Moving data.* Users in the room need to be able to move data among the various visualization applications that run on screens in the room and on laptops or PDAs that are brought into the workspace.
2. *Moving control.* To minimize disruption during collaboration sessions, any user should be able to control any device or application from his or her current location.
3. *Dynamic application coordination.* The specific applications needed to display data and analyze scenarios during team problem-solving sessions are potentially diverse. One company reported using over 240 software tools during a standard design cycle. Any number of these programs might be needed during a single meeting. The activities of each tool should coordinate with others as appropriate. For example, the financial impacts of a design change in a CAD program should automatically show up in a spreadsheet program that shows related information running elsewhere in the room.

Based on our experiences with the iRoom, we identified some key characteristics to be supported by the infrastructure and interfaces in an interactive workspace. Multiple devices (PDAs, workstations, laptops, and so forth) will be in simultaneous use in a workspace, with each chosen for its efficacy in accomplishing some specific task. There will also be heterogeneous software running on these devices, including both legacy and custom-built applications. All of these must be accessible to one another in a standard way so that the user can treat them as a uniform collection. This means that any software framework must provide cross-platform support. From the HCI perspective, interfaces must be customized to different displays and possibly to different input-output modalities, such as speech and voice.

The iRoom and Beyond: Evolution and Use of Deployed Environments

The iRoom project started with the first version of the Interactive Mural, a four-projector tiled display built in several stages from 1998 to 1999. It included a pressure-sensitive floor that tracked users in front of the display with one-foot accuracy. The pressure sensitive floor was used in some artistic applications but has not been duplicated in the iRoom. The first iteration of the iRoom was constructed in Summer 1999. Like the current version, it had three smart boards and the iTable, but it had a standard front-projected screen instead of the interactive mural at the front of the room.

Perhaps the biggest mistake we made in constructing the first iRoom was in planning the cabling. It might seem like an obvious thing in retrospect, but the number of cables needed to connect mouse, keyboard, video, networking, and USB devices quickly escalated, leaving a tangle of cables that were not quite long enough. For the second version of iRoom, we learned from our mistakes and made a careful plan of cable routes and lengths in advance. We made sure to label both ends of every cable with what they were connecting.

In Summer 2000, we integrated the Interactive Mural at the front of the iRoom, requiring a reconfiguration of the entire workspace. During the remodel, we introduced more compact light-folding optics for the projectors on the smart boards and did a better job of running the cables for the room. We added a developer lab adjacent to the room along with a sign-in area that holds mobile devices and a dedicated machine that could be used for room control. We configured the developer station with a KVM (keyboard-video-mouse) switch so that all of the iRoom PCs can be accessed from any of four developer stations. Figure A shows the floor plan of the second version of iRoom. One of the big headaches in building the second version of iRoom was dealing with projector alignment and color calibration.[1]

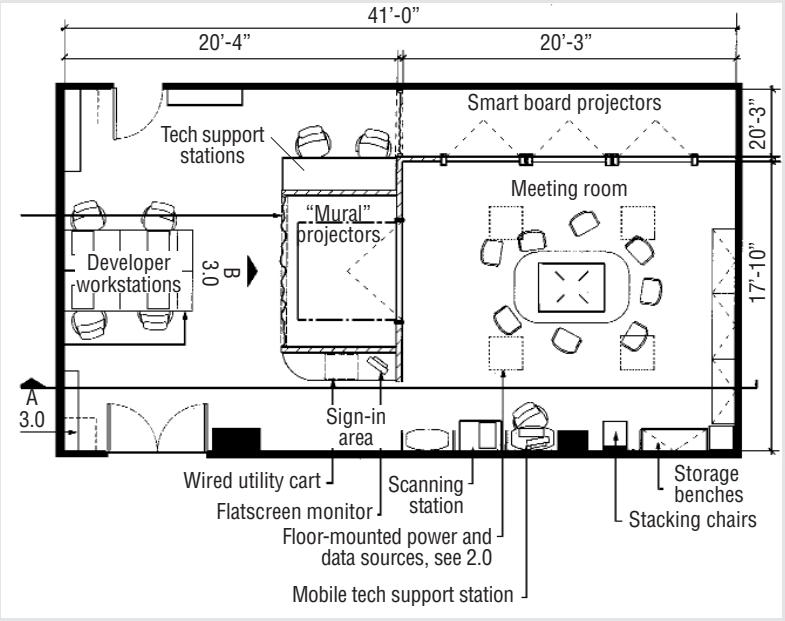


Figure A. Floor plan and behind-the-scenes look at the second version of iRoom.

Since building the second version, Interactive Workspaces technology has been deployed at six more locations around our campus. Through various collaborations, Interactive Workspaces group software is now being used in iRooms in Sweden and Switzerland. The i-Land[2] group has also done some work that uses the Event Heap in conjunction with their own software framework.

REFERENCES

1. M.C. Stone, "Color and Brightness Appearance Issues in Tiled Displays," *IEEE Computer Graphics & Applications*, vol. 21, no. 5, Sept./Oct. 2001, pp. 58–66.
2. N. Streitz et al., "i-LAND: An interactive Landscape for Creativity and Innovation," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 99), ACM Press, New York, 1999, pp. 120-127.

Furthermore, unlike a standard PC, an interactive workspace by its nature has multiple users, devices, and applications all simultaneously active. On short time scales, the individual devices might be turned off, wireless devices might enter and exit the room, and pieces of equipment might break down for periods of minutes, hours, or days. On longer time-scales, workspaces will incrementally evolve rather than be coherently designed and instantiated once and for all. While providing for these dynamic changes, interactive workspaces must also work with a minimum of administration if they are to be widely deployed. It is not realistic to expect a full-time system administrator to keep a workspace running, so we need to anticipate failure as a common case rather than as an exception.[2] The system must provide for quick recovery either automatically or through a simple set of user steps.

iROS meta-operating system

For any real-world system to support the modalities and characteristics just described, the system infrastructure must mirror the

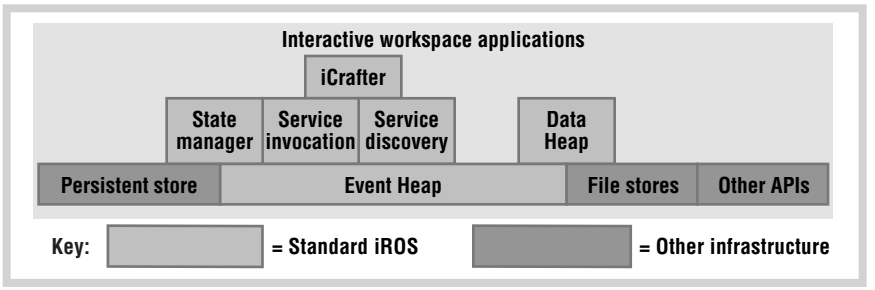


Figure 2. The iROS component structure.

applications and human-computer interfaces written on top of it. In addition, human-computer interfaces must consider the underlying system's properties to ensure that they are not too brittle for use in real-world situations. We call our system infrastructure the Interactive Room Operating System (iROS). It is a meta-OS that ties together devices that each have their own low-level OS. In designing the system, we kept the boundary principle[2] in mind. The boundary principle suggests that ubiquitous computing infrastructure must allow interaction between devices only within the bounds of the local physical space—in our case, an interactive workspace.

The three iROS subsystems are the Data Heap, iCrafter, and the Event Heap. They are designed to address the three user modalities of moving data, moving control, and dynamic application coordination, respectively. Figure 2 shows how the iROS components fit together. The only system that an iROS program must use is the Event Heap, which provides for dynamic application coordination and forms the underlying communication infrastructure for applications in the interactive workspace.

iROS subsystems

Given the heterogeneity in interactive workspaces and the likelihood of failure in individual devices and applications, it is important that the underlying coordination mechanism decouple applications from one another as much as possible. Doing so encourages applications to be less dependent on one another, which tends to make the overall system less brittle and more stable. We derive the Event Heap[5] coordination infrastructure for iROS from a tuplespace model,[6] which offers inherent decoupling.

The Event Heap stores and forwards messages known as events, each of which is a collection of name-type-value fields. It provides a central repository to which all applications in an interactive workspace can post events. An application can selectively access events on the basis of pattern matching fields and values. One key extension we made to tuplespaces was to add expiration to events, which allows unconsumed events to be automatically removed and provides support for soft-state through beaconing. Applications can interface with the Event Heap through several APIs, including Web, Java, and C++. The Event Heap differs from tuplespaces in several other respects that make it better suited for interactive workspaces.[5]

The Data Heap facilitates data movement by allowing any application to place data into a store associated with the local environment. The data is stored with an arbitrary number of attributes that characterize it. By using attributes instead of locations, applications don't need to worry about which specific physical file system stores the data. The Data Heap stores format information, and, assuming it loads the appropriate transformation plug-ins, it will automatically transform the data to the best format supported by retrieving applications. If a device only supports JPEG, for example, the Data Heap will automatically extract and convert a retrieved PowerPoint slide into that image format.

The iCrafter system[7] provides a system for service advertisement and invocation, along with a user interface generator for services. iCrafter services are similar to those provided by systems such as Jini,[8] except that invocation happens through the Event Heap. The interface manager service lets users select a service to control and then automatically returns the best interface for the user's device. iCrafter communicates directly with the services through the Event Heap. When a custom-designed interface is available for a device-service pair, the iCrafter system sends it. Otherwise, a more generic generator renders the interface into the highest quality type supported on the device. Generation happens with interface templates that are automatically customized according to the local environment's characteristics. If room geometry is available, for example, a light controller can show the actual positions of the lights on a graphical representation of the workspace. Templates also let users combine multiple services in a single interface.

General principles

iROS applications do not communicate directly with one another; instead, they use indirection through the Event Heap. This helps avoid highly interdependent application components that could cause each other to crash. All of the iROS systems decouple applications referentially with information routed by attribute rather than recipient name. Attribute-based naming is also used, among other places, in the Intentional Naming system.[9] The Event Heap and Data Heap also decouple applications temporally, allowing applications to pick up messages generated before they were running or while they were crashed and restarting.

In our design, we treat failure as a common case. When something breaks, the system can simply restart it. Clients automatically reconnect, so the Event Heap server, interface manager, and Data Heap server can all be restarted without interfering with applications other than during the period when connectivity is lost. Thus, any subset of machines malfunctioning in the workspace can be restarted. Any important state that might be lost during this process is either stored in persistent form in the Data Heap or is beaconed as soft-state as the clients come back online.

Because the Web is popular, a great deal

of technology has been developed and deployed that uses browsers and HTTP. We tried to leverage this technology wherever we could in the iROS system. The system supports moving Web pages from screen to screen, event submission through URLs and forms, and automatic HTML UI generation through iCrafter.

Human-computer interaction

In designing the interactive aspects of the iRoom, our goal has been to let the user remain focused on the work being done rather than on the mechanics of interaction. The HCI research has included two main components: developing interaction techniques for large wall-based displays and designing "overface" capabilities to provide access and control to information and interfaces in the room as a whole.

Our primary target user setting is one that we call an open participatory meeting. In this setting, a group of 2 to 15 people works to accomplish a task, usually as part of an ongoing project. People come to the meeting with relevant materials saved on their laptops or on file servers. During the meeting there is a shared focus of attention on a primary display surface, with some amount of side work that draws material from the shared displays and can bring new material to it. In many cases, a facilitator stands at the board and is responsible for overall activity flow. Other participants might also present something during the meeting. These meetings can at times include conventional presentations, but our goal is to facilitate interaction among participants.

Examples of such meetings conducted in the iRoom include our own project group meetings, student project groups in courses, construction management meetings, brainstorming meetings by design firms, and training-simulation meetings for school principals.

Large high-resolution displays

We were initially motivated to take advantage of interaction with large high-resolution displays. In a meeting, the presenter or facilitator focuses on the board's contents and on the other participants. Using a keyboard is distracting, so we designed methods for direct interaction with a pen or with direct touch on the board. The interactive mural is too large for today's touch-screen technologies, so we tested both laser and ultrasound technologies[10] as input devices. The current system uses an eBeam ultrasonic pen augmented with a button to distinguish two modes of operation, one for drawing and one for commands. The eBeam system does not currently support multiple simultaneous users.

We wanted to combine the benefits of two research threads in our interface: whiteboard functionality for quick sketching and GUI functionality for applications. We developed the PostBrainstorm interface[11,12] to provide a high-resolution display that has the ability to intermix direct marking, image control, 3D rendering, and arbitrary desktop applications. Our key design goal was to provide fluid interaction that would not divert user focus from person-to-person interaction. This goal led to developing several new mechanisms:

- FlowMenu is a contextual pop-up menu system that combines the choice of an action with parameter specification in a single pen stroke, which makes it possible to avoid interface modes that can distract users not devoting their full attention to the interface.[13] Because the menu is radial rather than linear, multilevel operations can be learned as a single motion path or gesture; in many cases the user does not even need to look at the menu to select an action.
- ZoomScape is a configurable warping of the screen space that implicitly controls an object's visible scale. The object retains its geometry while being scaled as a whole. In our standard configuration, the top quarter of the screen reduces the object size. An object can be moved out of the main area of the screen and reduced, providing a fluid mechanism to manage screen real estate without requiring explicit commands.
- Typed Drag-and-Drop is a handwriting recognition system that runs as a background process, leaving the digital ink and annotating it with the interpreted characters. Through FlowMenu commands, you can specify a sheet of writing to have a desired semantic and then drag it onto the target object to have the intended effect. This provides a crossover between simple board interaction and application-specific GUI interactions.

Several groups of industrial designers from two local design firms (IDEO and Speck-Design) tested the system. Their overall evaluation was positive[11] and alerted us to specific areas needing improvement. In addition to experimenting with these facilities on the high-resolution interactive mural, we ported them to a standard Windows systems and used them on the normal touch screens in the iRoom.

Room-based cross-platform interfaces

One obvious advantage of working in a room-based environment is that people share a common model of where devices are positioned, which they can use as a convenient way of identifying them. Our room controller (see Figure 3) uses a small map of the room to indicate the lights, projectors, and display surfaces. We use simple toggles and menus associated with objects in the map to switch video inputs to projectors and to turn lights and projectors on or off.

Initial versions of this controller were built as standard GUI applications, which could only run on some systems. We broadened their availability to a wider range of devices by providing them as Web pages (using forms) and as Web applets (using Java). Our later research generalized the process with iCrafter.[7] The room-control system stores the geometric arrangement of screens and lights in the room in a configuration file and will automatically provide controllers on any device supporting a UI renderer available through iCrafter. Figure 3 shows examples for several devices.

In addition to providing environment control, the same room control interface serves as the primary way to move information onto displays. The user indicates an information object (URL or file), the appropriate application to display it, and the display on which it should appear using the interface on their device. The user actions generate an event that is picked up by a dae-

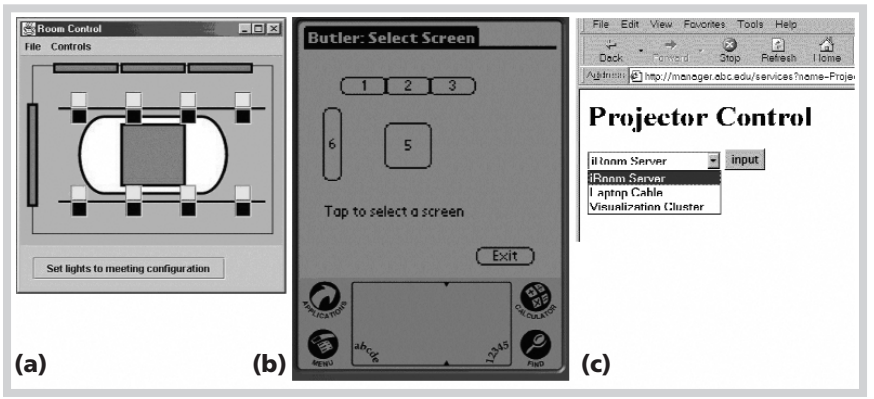


Figure 3. Examples of iCrafter-generated control interfaces: (a) Java Swing, (b) Palm handheld, and (c) HTML.

mon running on the target machine, which then displays the requested data.

Room-based input devices

In an interactive workspace, physical input devices belong to the space rather than a specific machine. We implemented an overhead scanner based on a digital camera. This scanner lets users digitize sketches and other material placed in a certain area of the table. This provided an alternative to sketching on tablet computers, which had the wrong feel when used by a team of brainstormers. The overhead scanner provides a method to bring traditional media into the space in a manner that has low cognitive overhead.

In addition to the overhead scanner, we introduced other devices, such as a bar-code scanner and simple wireless input devices, such as buttons and sliders. We used the bar-code scanner, for example, to implement a system similar to the BlueBoard system.[14] When the bar-code scanner posts an event, the application checks a table of codes registered to individual iRoom users; if there is a match, it posts the user's personal information space to one of the large boards. We can associate handheld wireless iRoom buttons with any actions through a Web-form interface. For example, a push on a particular button can bring up a set of predesignated applications on multiple devices in the room.

Distributed application control

One aspect of the moving-control modality for interactive workspaces is a need for both direct touch interaction with the GUIs on the large screens and the ability for users standing away from the screens to control the mouse and enter text. While it is possible to walk up to the screen to interact or to request that the person at the screen perform an action on your behalf, both of these actions disrupt the flow of a meeting. Several previous systems have dealt with multiuser control of a shared device, often providing sophisticated floor-control mechanisms to manage conflicts. In keeping with our keep-it-simple philosophy, we created a mechanism called PointRight,[15] which provides key functionality without being intrusive.

With PointRight, any machine's pointing device can become a superpointer whose field of operation includes all of the display surfaces in the room. When a device runs PointRight, the edges of its screen are associated with other corresponding displays. The user simply continues moving the cursor off the edge of the local screen, and it moves onto one of the other screens, as if the displays in the room were part of a large virtual desktop. In addition to allowing this control through laptops, the room has a dedicated wireless keyboard and mouse that is always available as a general keyboard and pointer interaction device for all of the surfaces. For each active user, their keystrokes go to the machine on which their pointer is currently active.

Through calls to the Event Heap, interface actions in one application can trigger actions in another running on any of the machines in the workspace. The Center for Integrated Facility Engineering[16] employed this technique in a suite of applications developed for use in construction-management meetings. Figure 4 shows some of the application viewers that they constructed.

All applications communicate through the Event Heap and emit events in a common format and watch for events to which they can respond. Users can coordinate the applications by bringing them up on any of the displays in an interactive workspace. As users select and manipulate information in one of the viewers, corresponding information in the other viewers updates to reflect changes. Because the components are loosely coupled, the absence or disappearance of an event source or event receiver does not affect any of the application components currently in use.

The Smart Presenter system lets users construct coordinated presentations across the displays in an interactive workspace. Users create a script that determines the content to display. PowerPoint and Web pages are two of the supported formats, but we can also use any other format that the Data Heap supports. In addition to content, we can send any arbitrary event so it is easy to trigger lighting changes or switch video inputs to a projector during a presentation. Smart Presenter leverages the Data Heap to insure it can show any display in the workspace. In the iRoom, for example, the high-resolution front display, which only supports JPEG images, can still display PowerPoint slides because they are extracted and transformed for that display.

Future directions

Several topics require additional investigation, including security, adaptation, and bridging interactive workspaces. While providing many important attributes, the loose coupling model introduces some security concerns. The indirect communication makes all messages public, which makes it easy to adapt programs to work with one another through intermediation at the expense of privacy. For now, we have

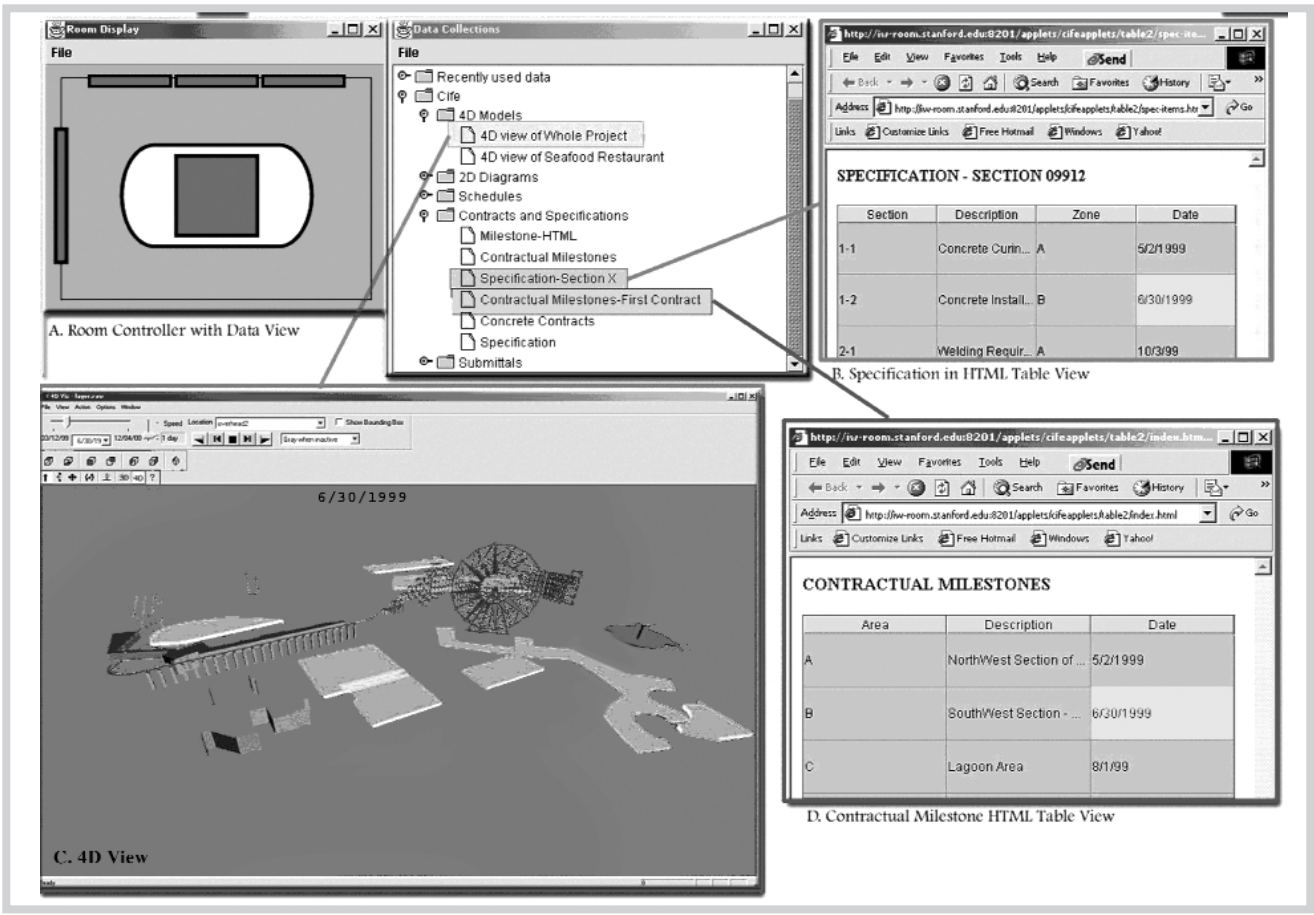


Figure 4. Some of the viewers in the suite that the Center for Integrated Facility Engineering developed for construction-management settings. Each would typically be run on its own display.

a firewall between the iRoom and the rest of the world and assume that users working together in a room have implicitly agreed to public communication. We are investigating the types of security that users need in interactive workspaces, with the hope of developing a social model for security that will in turn help us to define the appropriate software security protocols.

While our system tries to minimize the amount of time required to integrate a device into an interactive workspace, there is still overhead in configuring room geometries and specifying which servers to use. We plan to make it simpler to create and extend a workspace and to move portable devices between them. Users should only have to plug in a device or bring it into a physical space for it to become part of the corresponding software infrastructure. User configuration should be simple and prompted by the space. For example, the user might have to specify where in the room the device is located. The logical extension of this is to allow ad hoc interactive workspaces to form wherever a group of devices are gathered.

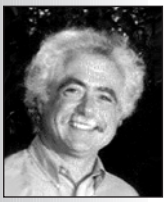
Allowing project teams in remotely located workspaces to work with one another is both interesting and useful. The main issues here include how to facilitate coordination between desired applications while insuring that workspace-specific events remain only in the appropriate location. For example, sending an event to turn on all lights should probably remain only in the environment where it was generated. As we extend our research to multiple linked rooms and remote participants, we will use observations of users working in these linked-spaces to determine how much structure we need to add. We are driven not by what is technically possible, but by what is humanly appropriate.

A key design philosophy for our project has been that the user should have a minimum of specific controls and modes to learn and remember, which means that the interface should take advantage of natural mappings to the physical structure. Although it would be an overstatement to say that the interface has become completely intuitive and invisible, we continue to make steps in that direction. As with all systems built in relatively new domains, and particularly with systems that involve user

the AUTHORS

Brad Johanson is a PhD candidate in the electrical engineering department at Stanford University and is one of the student leads in the interactive workspaces project. His research interests include genetic programming, computer networking, and computer graphics. He received a BA in computer science and a BS in electrical engineering and computer science from Cornell University, an MS in computer science from the University of Birmingham in England, and an MS in electrical engineering from Stanford University. Contact him at bjohanso@graphics.stanford.edu.

Armando Fox is an assistant professor at Stanford University. His research interests include systems approaches to improving dependability and system software support for ubiquitous computing. He received a BS in electrical engineering from MIT, an MS in electrical engineering from the University of Illinois, and a PhD in electrical engineering from the University of California at Berkeley. He is a member of the ACM and a founder of ProxiNet (now a division of PumaTech), which commercialized the thin client mobile computing technology he helped develop at UC Berkeley. Contact him at fox@cs.stanford.edu.



Terry Winograd is a professor of computer science at Stanford University, where he directs the interactivity laboratory and the program in human-computer interaction design. He is one of the principal investigators in the Stanford digital libraries project and the interactive workspaces project. His research interests include human-computer interaction design, with a focus on the theoretical background and conceptual models. Contact him at winograd@cs.stanford.edu.

interaction, it is difficult to come up with a quantitative measure of success.

We ran several experimental meetings, including brainstorming sessions by professional designers, construction of class projects built on the iROS system, training sessions for secondary school principals, construction-management experiments for a civil engineering research project, group writing in an English course, project groups from an interaction design course, and, of course, our own weekly group meetings. The overall results have been positive, with many suggestions for further development and improvement.

Comments from programmers who have appreciated how easy it is to develop applications with our framework are also encouraging. The adoption and spread of our technology to other research groups suggests that our system is meeting the needs of the growing community of developers for interactive workspaces. For more information on the Interactive Workspaces project and to download the iROS software, visit http://iwork.stanford.edu.

ACKNOWLEDGMENTS

We thank Pat Hanrahan, one of the founders of the project, for his support and efforts. The accomplishments of the Interactive Workspaces group are due to the efforts of too many to enumerate here, but you can find a complete list on the Interactive Workspaces Web site. DoE grant B504665, NSF Graduate Fellowships, and donations of equipment and software from Intel, InFocus, IBM, and Microsoft have supported the work described here.

REFERENCES

1. M. Weiser, "The Computer for the 21st Century," *Scientific American*, vol. 265, no. 3, 1991, pp. 66–75.
2. T. Kindberg and A. Fox, "System Software for Ubiquitous Computing," *IEEE Pervasive Computing*, vol. 1, no. 1, Jan./Feb. 2002, pp. 70–81.
3. B. Brumitt et al., "EasyLiving: Technologies for Intelligent Environments," *Proc. Handheld and Ubiquitous Computing 2nd Int'l Symp. HUC 2000*, Springer-Verlag, New York, 2000, pp. 12–29.
4. M.H. Coen et al., "Meeting the Computational Needs of Intelligent Environments: The Metaglue System," *Proc. 1st Int'l Workshop Managing Interactions in Smart Environments*, 1999.
5. B. Johanson and A. Fox, "The Event Heap: An Coordination Infrastructure for Interactive Workspaces," to be published in *Proc 4th IEEE Workshop Mobile Computer Systems and Applications* (WMCSA 2002), IEEE Press, Piscataway, N.J., 2002.
6. N. Carriero and D. Gelernter, "Linda in Context (Parallel Programming)," *Comm. ACM*, vol. 32, no. 4, 1989, pp. 444–458.
7. S. Ponnekanti et al., "ICrafter: A Service Framework for Ubiquitous Computing Environments," *Proc. Ubicomp 2001*, Springer-Verlag, New York, 2001, pp. 256–272.
8. J. Waldo, "The Jini Architecture for Network-Centric Computing," *Comm. ACM*, vol. 42, no. 7, 1999, pp. 76–82.
9. W. Adjie-Winoto et al., "The Design and Implementation of an Intentional Naming System," *Operating Systems Rev.*, vol. 33, no. 5, Dec. 1999, pp. 186–201.
10. X.C. Chen and J. Davis, "LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays," *Displays*, vol. 22, no. 1, Mar. 2002.
11. F. Guimbretière, *Fluid Interaction for High Resolution Wall-Size Displays*, PhD dissertation, Computer Science Department, Stanford Univ., Calif., 2002.
12. F. Guimbretière, M. Stone, and T. Winograd, "Fluid Interaction with High-Resolution Wall-Size Displays," *Proc. of the ACM Symp.*, ACM Press, New York, 2001, pp. 21–30.
13. J. Raskin, *The Humane Interface: New Directions for Designing Interactive Systems*, Addison Wesley, Reading, Mass., 2000.
14. D. Russell and R. Gossweiler, "On the Design of Personal & Communal Large Information Scale Appliances," *Proc. Ubicomp 2001*, Springer-Verlag, New York, 2001, pp. 354-361.
15. B. Johanson, G. Hutchins, and T. Winograd, "PointRight: A System for Pointer/Keyboard Redirection Among Multiple Displays and Machines," tech. report CS-2000-03, Stanford Univ., 2000; http://graphics.stanford.edu/papers/pointri ht.
16. K. Liston, M. Fischer, and T. Winograd, "Focused Sharing of Information for Multidisciplinary Decision Making by Project Teams," *Proc. ITcon*, vol. 6, 2001, pp. 69–81.