# The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms

**Article** *in* IEEE Pervasive Computing · May 2002

DOI: 10.1109/MPRV.2002.1012339 · Source: IEEE Xplore



**3 authors**, including:

Brad Johanson
Tidebreak, Inc.
**29** PUBLICATIONS **1,903** CITATIONS

SEE PROFILE

Armando Fox
University of California, Berkeley
**199** PUBLICATIONS **26,725** CITATIONS

SEE PROFILE

# The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms [Version #2, 4/11/02]

Pervasive Computing Magazine Special Issue on Systems

Brad Johanson, Armando Fox, Terry Winograd

Stanford University, Stanford, CA

{bjohanso,fox,Winograd}@graphics.Stanford.edu

## Introduction

The Interactive Workspaces project was started at Stanford University in mid-1999 as an extension of a project to investigate interaction with large high resolution displays. It was initially set up in a busy laboratory where the device proved to be no more than a curiosity, since it could not be practically 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 through its embedding in a ubiquitous computing environment that provided for sustained realistic interactive use. The interactive workspaces project was founded to investigate the design and use of rooms containing one or more large displays with the ability to integrate portable devices and to create applications integrating the use of multiple devices in the space.

The idea of ubiquitous computing [18] is broad, encompassing 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 [10].
- Emphasizing the use of large interactive walk-up displays, some using touch interaction.
- Collaborating with other research groups to construct "non-toy" applications in design and engineering.

We have 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 (HCI) in the space. Further, we have assisted outside groups in using our technology to construct application suites that address problems in their own domains, and deployed our software in production environments. This paper gives a broad overview of these activities and the insights we have gained through the process.

## Project Overview, Goals, Contributions

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

- **Practice what we preach.** From the beginning we have used the iRoom as our main project meeting room and have 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 co-location.** There is a long history of research on computer supported cooperative work 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.
- **Reliance on social conventions.** Many projects have attempted to make an interactive workspace "smart" (usually called an *intelligent environment*) [2, 5]. Rather than have the room react to users, we have chosen to focus on providing the affordances necessary for a group to adjust the environment as they proceed with their task. In other words, we have set our *semantic Rubicon* [10] such that users and social conventions take responsibility for actions, and the system infrastructure is responsible for providing a fluid means to execute those actions.
- **Wide applicability.** Rather than investigating systems and applications just in our specific space, we decided to investigate software techniques that would also apply in differently configured workspaces. Our goal is to provide a framework similar to the device-driver model, window-manager system, and look and feel guidelines for PCs. We want to create standard abstraction and application design methodologies that apply to any interactive workspace.
- **Keep it simple.** At both the interface and software development levels, we try to keep things simple. On the human-interface side, we face a fundamental tradeoff in interaction design between the necessity of supporting diverse hardware and software and the need to provide an interface simple enough that people will use it. The system must remain accessible to the non-expert intermittent users that can be expected in an interactive workspace. On the software development side, we try 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.

**Figure 1 - A View of the Interactive Room (iRoom)**

## *The iRoom*

The iRoom, short for Interactive Room, is our second generation prototype interactive workspace (see Figure 1). Several other iRooms have been created both at Stanford and elsewhere (see sidebar). The iRoom contains three touch sensitive white-board sized displays along the side wall, and a custom-built 9 megapixel, 6' diagonal display with pen interaction called the interactive mural built into the front wall. In addition, there is a table with a built in 3' x 4' display that was custom designed to look like a standard conference room table. The room also has cameras, microphones, wireless LAN support, and a variety of wireless buttons and other interaction devices.

## *Common Usage Modalities*

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 characteristics of tasks:

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 laptops or PDAs that are brought into the workspace.
2. **Moving Control.** To minimize disruptions during collaboration sessions, any user should be able to control any device or application from their current location. One specific need is a means of providing mouse and keyboard control for GUIs on machines across the room from the user through their local laptop or PDA.
3. **Dynamic Application Coordination.** The specific applications that are 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), and any number of these programs may 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 showing related information running elsewhere in the room.

### *Characterizing the Environment*

Based on our experiences with the iRoom, we identified some key characteristics to be supported by the infrastructure and interfaces in an interactive workspace:

**Heterogeneity:** A variety of different devices (PDAs, workstations, laptops, etc.) will be in use in the workspace, each chosen for their 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 need to 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 need to be customized to different sized displays, and possibly different input/output modalities such as speech and voice.

**Multiplicities:** Unlike a standard PC where a single user and set of input and output devices provide interaction with the machine, an interactive workspace by its nature has multiple users, devices, and applications all simultaneously active.

**Dynamism:** Interactive workspaces will be dynamic. On short time scales, individual devices may be turned off, wireless devices will enter and exit the room, and pieces of equipment may break down for periods of minutes, hours or days. On longer time scales workspaces will incrementally evolve rather than being coherently designed and instantiated once and for all. In providing for this dynamic change, interactive workspaces will only be widely deployed if they "just work." It is not realistic to expect a full-time system administrator to keep a workspace running, and at the same time users must be allowed to integrate even failure prone devices. Thus, failure needs to be anticipated as a common case, rather than an exception [10], and the system must provide for quick recovery, either automatically or via a simple set of steps for the user.

## The iROS Meta-Operating System

For any real world system to support the modalities and characteristics just described, systems infrastructure and human interface issues must be looked at together. The system infrastructure must mirror the applications and human-computer interfaces that will be written on top of it, and human-computer interfaces must take into account the properties of the underlying system to insure that they are not too brittle for use in real world situations. This section discusses the system infrastructure we built, while the next presents our human-computer interaction research.

Our system infrastructure is called the Interactive Room Operating System (iROS). It is a meta-operating system or middleware infrastructure tying together devices that each have their own low-level operating system. In designing it we have kept in mind the boundary principle [10] which states that ubiquitous computing infrastructure needs to allow interaction between devices only within the bounds of the local physical space, in our case an interactive workspace. A running iROS system is therefore associated with a specific physical interactive workspace and support the human-computer interaction needs of applications which will be running therein.

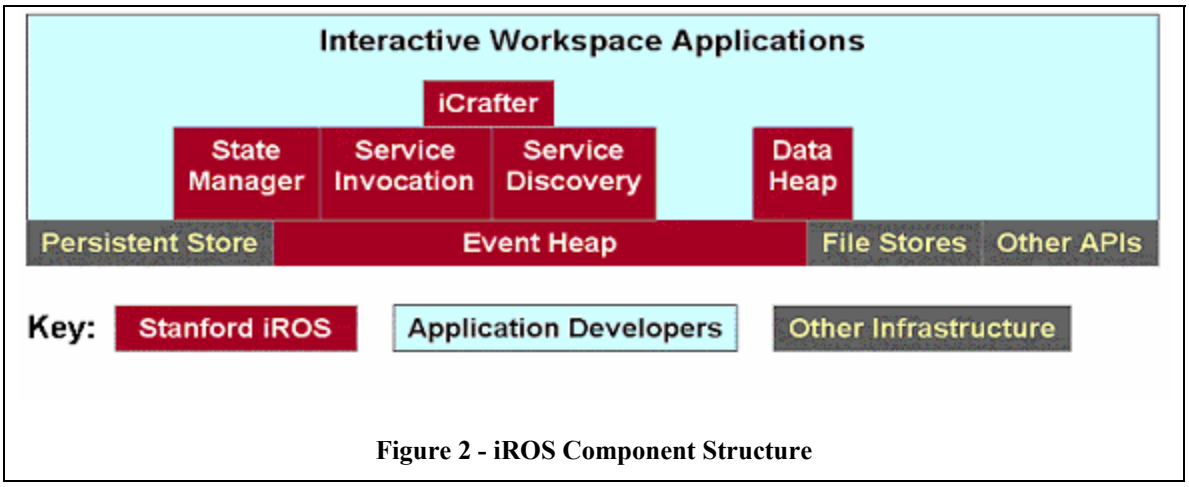



**Figure 2 - iROS Component Structure**

The three iROS sub-systems 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 in addition to providing for dynamic application coordination is also the underlying communication infrastructure for applications within an interactive workspace.

## *iROS Sub-systems:*

### The Event Heap

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. This encourages applications to be written which are less dependent on one another, thereby making the overall system less brittle and more stable. The Event Heap [8] coordination infrastructure for iROS is derived from a tuplespace model [3], 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 based on a pattern match over fields and values, and can retrieve either destructively or non-destructively. One key extension we made to tuplespaces was to add expiration to events. This 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++. There is a standard TCP/IP protocol, making it easy to create clients for other platforms. The Event Heap differs from tuplespaces in several other respects which make it better suited for interactive workspaces. The Event Heap is presented in greater detail in [8].

### The Data Heap

The Data Heap facilitates data movement in an interactive workspace. It allows 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, and can be retrieved by a query specifying attributes that must be matched. By using attributes instead of locations, applications don't need

to worry about which specific physical file system is being used to store the data. Format information is also stored in the Data Heap, and, assuming appropriate transformation plug-ins are loaded, data is automatically transformed to the best format supported by retrieving applications. If a device only supports JPEG, for example, a retrieved Power Point slide will automatically be extracted and converted into that image format.

### iCrafter

The iCrafter system [12] 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 [17], except that invocation is through the Event Heap, and soft-state beaconing is used instead of leases. The novel aspect of iCrafter is the interface manager service which provides a method for users to select a service or set of services to control, and then automatically returns the best interface to the service(s) for the users device. The interface iCrafter generates communicates directly with the services through the Event Heap. When a custom-designed interface is available for a device/service pair, it will be sent. Otherwise, a more generic generator will be used to render into the highest quality interface type supported on the device. Generation is done using interface templates that are automatically customized based on the characteristics of the local environment. For example, if room geometry is available, a light controller can show the actual positions of the lights on a graphical representation of the workspace. Templates also allow multiple services to be combined together in a single interface—all lights and projectors in a workspace can be put together, for example.

## *General Principles*

Some common principles run throughout the iROS system:

**Decoupling to Make System More Robust:** iROS applications do not communicate directly with one another, but use indirection through the Event Heap. This helps avoid highly interdependent application components, which have the potential to 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 [1]). The Event Heap and Data Heap also decouple applications temporally due to persistence, allowing applications to pick up messages generated before they were running, or while they were crashed and restarting.

**Modular Restartability:** In our design, failure is treated as a common case, so when something breaks it can simply be restarted. 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 that are malfunctioning in the workspace can be restarted in any order to get it back up and running. 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 which is quickly regenerated as clients come back up.

**Leveraging the Web:** Due to the popularity of the World Wide Web, a great deal of technology has been developed and deployed that utilizes browsers and the HTTP protocol. We tried to leverage that wherever we could in the iROS system. We support both movement of web pages from screen to screen, event submission via URLs and form pages, and automatically generated HTML UIs via iCrafter.

# Human-Computer Interaction: Issues and Examples

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

Our primary target user setting is one which we call an "open participatory meeting." In this setting, a group (2 to ~15 people) works together to accomplish a task, usually as part of an ongoing project. People come to the meeting with relevant materials on their laptops or saved on file servers, in a variety of formats, for different applications that will be used as part of the meeting. 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 flow of the activities, but other participants may also present during the course of the meeting. These meetings at times include conventional presentations, but our thrust is to facilitate interaction among participants in the room.

Examples of such meetings that have been 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.

## *Interaction with large high-resolution displays*

The initial motivation for the iRoom was to take advantage of interaction with large high-resolution displays, such as the Interactive Mural. The attention of a presenter or facilitator in a meeting is focused on the contents of the board and on the other participants. Any use of a keyboard is a distraction, so we have designed methods for direct interaction with a pen and with direct touch on the board.

The Interactive Mural is too large for today's touch screen technologies, and we have tested both laser and ultrasound technologies [4] as input mechanisms. 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 handwriting and GUI functionality for applications. We developed the PostBrainstorm interface [7] [6] to provide a high-resolution display with the ability to intermix direct marking, control of images, 3D rendering, and arbitrary desktop applications. The key design goal was to provide "fluid interaction" which does not divert the users focus from person-to-person interactions in a meeting. This goal led to the development of several new mechanisms:

- **FlowMenu:** a contextual pop-up menu system that combines the choice of an action with parameter specification in a single pen stroke. This makes it possible to avoid interface modes which can distract users not devoting their full attention to the interface (see [13] for discussion). Because the menu is radial rather than linear, multi-level operations can be

learned as a single motion path or gesture, so in many cases the user does not even need to look at the menu to select an action.

- **ZoomScape**: a configurable "warping" of the screen space so that the visible scale of an object is implicitly controlled by where it is moved. The object retains its geometry while being scaled as a whole. In our standard configuration, the top quarter of the screen is a reduction area, in which objects are one-quarter size. An object can be moved out of the main area of the screen and reduced all in one pen stroke, with a smooth size transition as it goes through a boundary area. This provides a fluid mechanism for screen real-estate management without requiring explicit commands to change size, iconify, etc.
- **Typed Drag-and-Drop**: Handwriting on the screen is recognized by a background process, leaving the digital ink and annotating it with the interpreted characters. Through FlowMenu commands, a sheet of writing can be specified to have a desired semantic (e.g., the name and value of a property to be associated with an object) and then dragged onto the target object to have the intended effect. This provides a crossover between simple board interaction (hand drawn text) and application-specific GUI interactions.

The overall system was tested in actual use by several groups of industrial designers from two local design firms (IDEO and SpeckDesign). Their overall evaluation of the facility was quite positive [6] and provided us with a number of specific areas for improvement.

In addition to experimenting with these facilities on the high-resolution interactive mural, we have ported them to standard Windows systems, and have made use of them on the normal touch-screens in the iRoom.

## *Overface*

The overface, which provides access and control to the interactive workspace as a whole, needs to provide a variety of functions to users of the room:

- Controlling the environment (lights, projectors, display sources, …)
- Posting information of all kinds from anywhere onto any of the display surfaces
- Controlling applications running on any of the display surfaces

Based on our observed need for moving control, these should all be achievable from any of the devices in the room. Several prototypes have been developed, as described in the following sections.

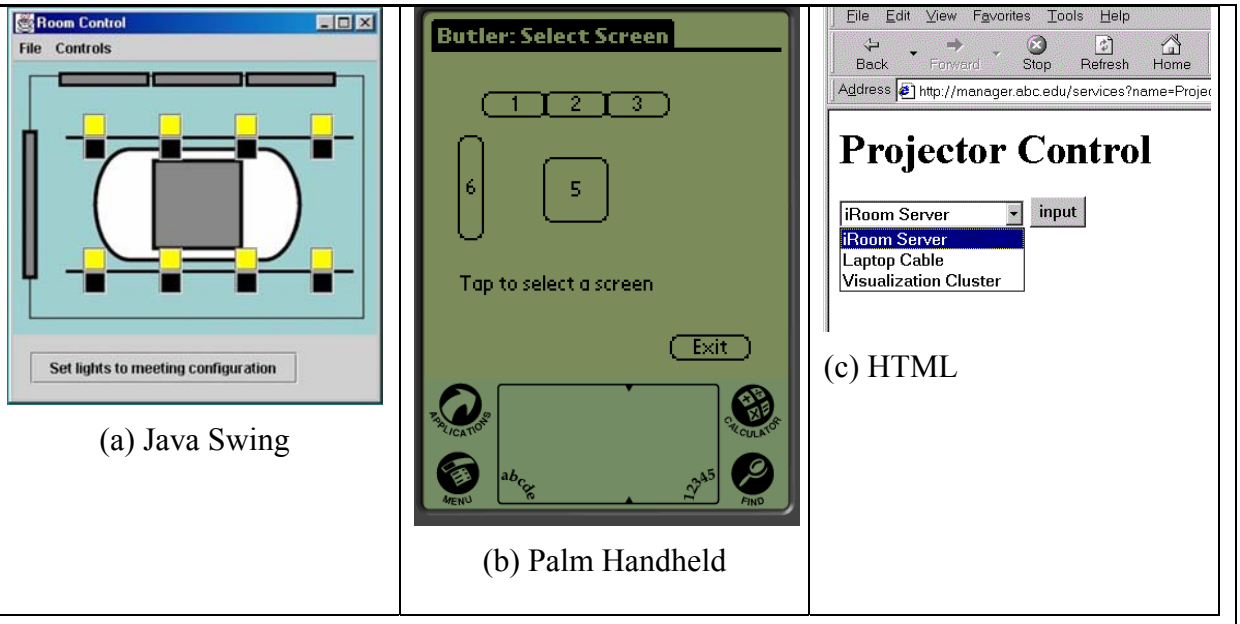
## *Room-based cross-platform interfaces*

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. Simple toggles and menus associated with objects in the map can be used to switch video inputs to projectors as well as 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 further with iCrafter [12], which was discussed earlier.

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 gives examples for several devices.




(a) Java Swing

(b) Palm Handheld

(c) HTML

**Figure 3 – iCrafter Generated Light and Projector Control Interfaces**

In addition to providing environment control, the same room control interface serves as the primary way of moving 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. On platforms that support it, drag-and-drop can be used. The user actions generate an event that is picked up by a daemon 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. One example that we have implemented is an overhead scanner which is based on a digital camera. It allows sketches and other material to be digitized when they are placed in a certain area of the table. Crop marks can be used to select the exact region, and when the camera is triggered, an image of the object is placed on the specified screen. This provided an alternative to tablet computers for sketching, which were found to have the wrong "feel" when used by a team of brainstormers. The overhead scanner provides a method of bringing traditional media into the space in a manner that has low cognitive overhead.

In addition to the overhead scanner, we have introduced other devices such as a bar code scanner and simple wireless input devices, such as buttons and sliders. These post events on the Event Heap, which are available for any program to use, and can therefore be easily adapted to

different functions. For example, the bar code scanner was used to implement a system similar to the BlueBoard system [14]. When the barcode scanner posts an event, the application checks a table of codes registered to individual iRoom users, and if there is a match, it posts the user's personal information space to one of the large boards. Handheld wireless iRoom buttons can be associated with any actions through a web-form interface. For example, a push on a particular button can bring up a set of pre-designated applications on multiple devices in the room, to set up a meeting context.

## *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 the person at the screen to perform an action on your behalf, both of these disrupt the flow of a meeting. A number of previous systems have dealt with multi-user 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 [9], which provides the key functionality without being intrusive.

### PointRight

With PointRight, any machine's pointing device can become a "super pointer" whose field of operation includes all of the display surfaces in the room, as well as the machine it is on. When a device runs PointRight, the edges of its screen are associated with corresponding other displays. So 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 which 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. The system also tracks state of which machines and displays are on, and which machine is providing video to each screen. It automatically routes the mouse control to the visible machine for each active display. This allows, for example, interaction with laptops being displayed on the touch screens directly through the touch screens. Currently nothing special is done to handle multiple pointers active on a screen—cursor update events are simply time-multiplexed by the OS. This works well in practice since social protocol quickly resolves which user will get to be active on the screen in contention.

### CIFE Suite: Example of Dynamic Application Coordination

Through calls to the Event Heap, interface actions within one application can trigger actions within another running on any of the machines in the workspace. This has been employed in a suite of applications developed by The Center for Integrated Facility Engineering (CIFE) [11] for use in construction management meetings. Figure 4 shows some of the application viewers that they have constructed.

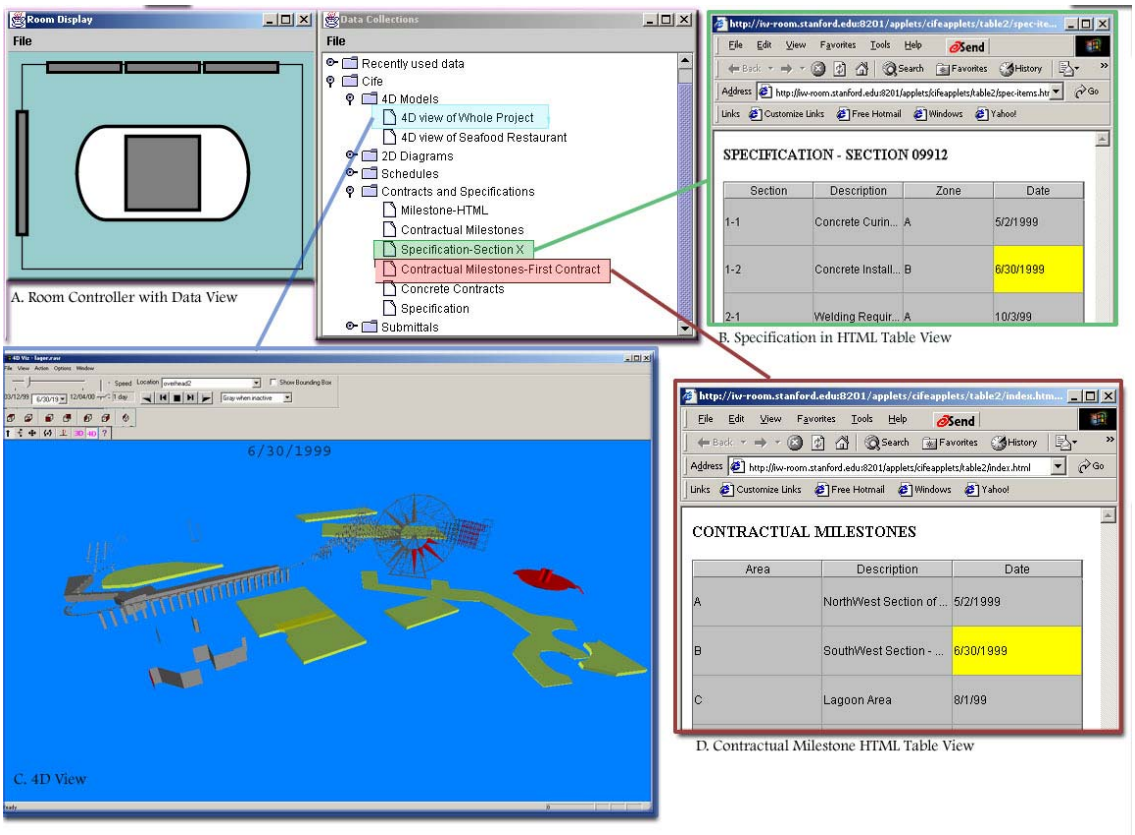



**Figure 4 – Some of the Viewers in the CIFE Suite**

All of the applications are essentially standalone, and communicate through the Event Heap. Applications emit events in a common format and match 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 becomes selected or updates to reflect changes. Since 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.

## Smart Presenter

The Smart Presenter system allows the construction of coordinated presentations across the displays in an interactive workspace. Users create a script of which content to display on what displays for any point in the presentation. PowerPoint and web pages are two of the default supported formats in the system, but any data format supported by the Data Heap may also be used. In addition to content, any arbitrary event may be sent, so it is straightforward to trigger lighting changes or switch video inputs to a projector during a presentation.

Smart Presenter leverages the Data Heap heavily to insure that any content can be shown on any display in a workspace. In the iRoom, for example, the high-resolution front display, which only supports JPEG images, can still be used to display PowerPoint slides since they are extracted and transformed for that display.

## *Interaction philosophy: Minimal interaction, implicit structure*

A key design philosophy for our project has been: the user should have a minimum of specific controls and modes to learn and remember, and the interface should take advantage of natural mappings to the physical structure. Examples include the use of a physical map for controllers, the ZoomScape mechanism for scaling images based on location, the overhead camera scanner interface for entering sketches and other visual material, and cross-linking mechanisms that enable actions in one structure (such as a project plan) to automatically trigger actions in another (the CAD drawing). 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.

## SIDEBAR 1:

# The iRoom and Beyond: Evolution and Usage of Deployed Environments [SIDEBAR]

## Early Work: The Interactive Mural

The 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, which tracked users in front of the display with 1' accuracy. The pressure sensitive floor was used in some artistic applications, but has not been duplicated in the iRoom.

## iRoom v1

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 the construction of the first iRoom was in planning the cabling. It seems like an obvious thing, but the number of cables needed to connect mouse, keyboard, video, networking, USB devices etc. quickly escalates, leaving a tangle of cables that are not quite long enough and going to unknown devices. For iRoom v2 we learned from our mistakes and made a careful plan of cable routes and lengths in advance, and made sure to label both ends of every cable with what they were connecting.

## iRoom v2

In Summer of 2000, the Interactive Mural was integrated as 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 over one-half mile of cables for the room. A developer lab adjacent to the room was added, along with a sign-in area that holds mobile devices and a dedicated machine that can be used for room control. The developer station has been configured with a KVM (Keyboard-Video-Mouse) switch so that all of the iRoom PCs may be accessed from any of four developer stations. The floor plan of iRoom 2 is shown in Figure 5.

One of the big headaches in building iRoom v2 was dealing with projector alignment and color calibration. More about this can be found in [15].

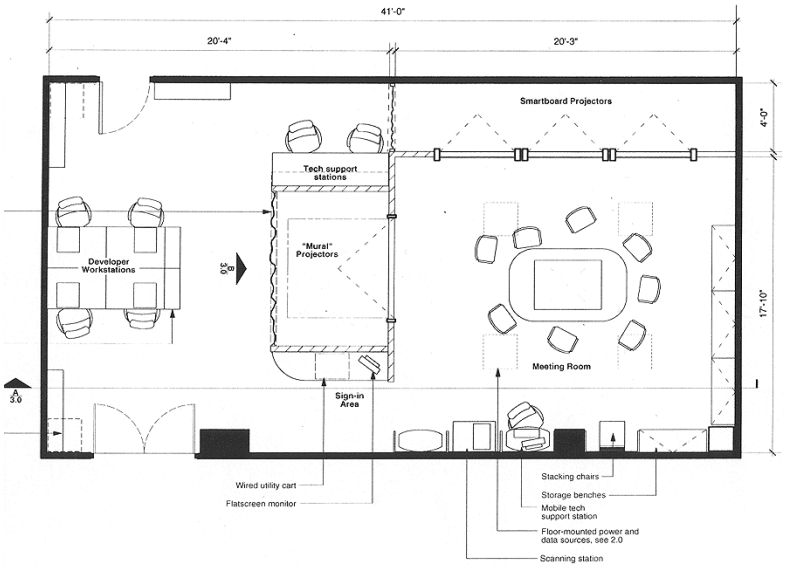



**Figure 5 - Floor Plan and Behind the Scenes of iRoom v2**

## The Proliferation of Interactive Workspaces

Since the building of iRoom v2, Interactive Workspaces technology has been deployed at six more locations around campus. Through various collaborations, Interactive Workspaces group software is now being used in iRooms in Sweden and Switzerland. The i-Land [16] group has also done some work which uses the Event Heap in conjunction with their own software framework.

## Related Projects

*NOTE TO THE EDITORS: We were advised by the reviewer to point to an article with an overview of related projects here. Unfortunately, we know of no such article. We were hoping that this issue might include articles from projects similar to ours, and would serve this function in the future. The projects we consider related are:*

- *Easy Living, Microsoft*
- *Intelligent Room, MIT*
- *i-Land Project, GMD-IPSI Darmstadt*
- *Gaia OS, UIUC*

*This section could just be a brief pointer to articles on any of these projects should they be in this issue. If not, we would be happy to write a brief blurb here, or this section could be removed.*

## Future Directions

Several topics call for further investigation. Three of the most prominent are security, adaptation, and bridging interactive workspaces.

While providing many important attributes, the loose coupling model also introduces some security concerns. The indirect communication makes all messages public. This makes it easy to adapt programs to work with one another through intermediation, but also brings up concerns of security and privacy. For now, we firewall the iRoom off from the rest of the world and assume that users working together in a room have implicitly agreed to public communication. We are undertaking an investigation of the types of security that users need in interactive workspaces, with the subsequent development of a social model for security that will in turn shape the software security protocols to be developed.

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 in order for it to become a part of the corresponding software infrastructure. User configuration should be simple and prompted by the space—for example the user might be requested 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.

As mentioned earlier, we have so far focused on co-located collaboration. Allowing project teams in remotely located interacted workspaces to work with one another is both interesting and useful. The main issues here are 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 the work to multiple linked rooms and remote participants, we will use observations of the work to determine how much additional structure needs to be added. We are driven not by what is technically possible, but what is humanly appropriate.

## Final Words

As with all systems being built in relatively new domains, and particularly with systems that involve user interaction, it is difficult to come up with a quantitative measure of success. We have had a number of experimental uses, including:

- Design brainstorming sessions by professional designers
- Construction of class projects built on the iROS system
- Training sessions for secondary school principals
- Construction management experiments as part of a Civil Engineering research project
- Group writing in a Stanford 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. Finally, the adoption and spread of our technology to other research groups (see sidebar) also 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 (including easy installers for Windows NT/2000/XP), please go to http://iwork.stanford.edu.

## Acknowledgments

We wish to 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 a complete list may be found on the Interactive Workspaces website. The work described has been supported by DoE grant B504665, by NSF Graduate Fellowships, and by donations of equipment and software from Intel Corp., InFocus, IBM Corp. and Microsoft Corp.

## References

1. Adjie-Winoto, W., *et al.*, *The design and implementation of an intentional naming system.* Oper. Syst. Rev. (USA), Operating Systems Review, 1999. **33**: p. 186-201.
2. Brumitt, B., *et al. EasyLiving: technologies for intelligent environments*. in *Handheld and Ubiquitous Computing Second International Symposium HUC 2000*. 2000. Bristol, UK: Berlin, Germany : Springer-Verlag, 2000.
3. Carriero, N. and D. Gelernter, *Linda in context (parallel programming)*. Communications of the ACM, 1989. **32**(4): p. 444-58.
4. Chen, X.C. and J. Davis, *LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays.* Displays, 2002. **22**(1).
5. Coen, M.H., *et al. Meeting the Computational Needs of Intelligent Environments: The Metaglue System*. in *MANSE99 : 1st International Workshop Managing Interactions in Smart Environments*. 1999. Dublin, Ireland.
6. Guimbretière, F., *Fluid Interaction for High Resolution Wall-size Displays*. Ph.D. Dissertation, Computer Science. 2002, Stanford, CA, USA: Stanford University. 140.
7. Guimbretière, F., M. Stone, and T. Winograd, *Fluid Interaction with High-resolution Wall-Size Displays.* UIST (User Interface Software and Technology): Proceedings of the ACM Symposium, 2001: p. 21-30.
8. Johanson, B. and A. Fox. *The Event Heap: An Coordination Infrastructure for Interactive Workspaces*. in *To appear in Proceedings of the 4th IEEE Workshop on Mobile Computer Systems and Applications (WMCSA-2002)*. 2002. Callicoon, New York, USA.
9. Johanson, B., G. Hutchins, and T. Winograd, *PointRight: A System for Pointer/Keyboard Redirection Among, Multiple Displays and Machines*. CS Tech Report, CS-2000-03. 2000, Stanford, CA: Stanford University. http://graphics.stanford.edu/papers/pointright/.

10. Kindberg, T. and A. Fox, *System Software for Ubiquitous Computing*, in *IEEE Pervasive Computing*. 2002. p. 70-81

11. Liston, K., M. Fischer, and T. Winograd, *Focused Sharing of Information for Multi-disciplinary Decision Making by Project Teams.* ITcon, 2001. **6**: p. 69-81.

12. Ponnekanti, S., *et al. ICrafter: A Service Framework for Ubiquitous Computing Environments*. in *Ubicomp 2001*. 2001. Atlanta, GA, USA.

13. Raskin, J., *The humane interface : new directions for designing interactive systems*. 2000, Reading, Mass.: Addison Wesley. xix, 233.

14. Russell, D. and R. Gossweiler. *On the Design of Personal & Communal Large Information Scale Appliances*. in *Ubicomp 2001*. 2001. Atlanta, GA, USA.

15. Stone, M.C., *Color and brightness appearance issues in tiled displays.* IEEE Computer Graphics and Applications, 2001. **21**(5): p. 58-66.

16. Streitz, N., *et al. i-LAND: An interactive Landscape for Creativity and Innovation*. in *ACM Conference on Human Factors in Computing Systems (CHI'99)*. 1999. Pittsburgh, PA, USA: ACM Press, New York, NY, USA.

17. Waldo, J., *The Jini architecture for network-centric computing.* Communications of the ACM, 1999. **42**(7): p. 76-82.

18. Weiser, M., *The computer for the 21st century.* Scientific American, 1991. **265**(3): p. 66-75.