iStuff: Searching For The Great Unified Input Theory

Jan Borchers, Rafael Ballagas
Stanford University
E-mail: {*borchers, ballagas*}*@cs.stanford.edu*

Maureen Stone
StoneSoup Consulting
E-mail: *stone@stonesc.com*

ABSTRACT
What will be the equivalent of mouse and keyboard, windows and icons--the ubiquitous components of the GUI in the post-desktop era? To explore this question, we have developed a toolkit and framework called *iStuff* that facilitates experimentation with multiple technologies, modalities, and metaphors for user interfaces in a ubicomp environment. Our domain is explicit interaction with a room sized environment consisting of displays of many sizes, plus support for wireless technology of various types, integrated using a common middleware. Our goal is to allow multiple, co-located users to fluidly interact with any of the displays and applications in the room, using for input any devices conveniently at hand.

INTRODUCTION
Today's desktop applications have essentially settled on simple variations of the WIMP interface, sometimes augmented with speech and audio. Projecting such desktop systems onto walls or hand-held devices quickly exposes some fundamental assumptions about display size and resolution that break down when the display is either too large or too small. It also exposes an even more fundamental assumption--that each display comes with its own pointing device and keyboard conveniently at hand.

The Stanford iRoom (Figure 1) combines wall-sized displays with portable devices of many types to create a shared, interactive workspace. For this class of environments, the software infrastructure group at Stanford has developed the iRoom Operating System, the *iROS*, a TCP-based middleware that allows multiple machines and applications to exchange information [iROS02]. The iROS supports communication through the *Event Heap*, a central server process that receives events from client applications in the room and redistributes them to the appropriate recipients. This creates a communications mechanism that extends the notion of an event queue to an entire interactive room, with multiple machines and users. It is designed specifically to be robust against failure, and to support easy restarting of arbitrary parts of the system (including the central Event Heap itself). The iROS is available in Open Source distribution from iwork.stanford.edu.

The machines in the iRoom run standard operating systems and applications, rather than primarily systems designed exclusively for the environment [Streitz99, Tandler01, Rekim99, Meyers98] Applications developed for the iRoom typically consist of suites of programs that combine their own UI's with linked interaction via the iROS. This approach allows for incremental deployment of complex systems, such as those developed for construction management [CIFE02].

From the first implementation of the iRoom, we have found it important to break the traditional tight binding between displays, machines and input devices. Users want to display information anywhere they can conveniently see it, using input devices that are conveniently at hand.

Figure 1. The interactive room (iRoom), showing its three SMART boards, interactive table, wireless keyboard and mouse, an other wireless input. The unlit screen on the left is part of the high-resolution, interactive "mural" [Mural01]

Ubiquitous Interaction in the iRoom
The first iRoom projects to address the problem of smoothly retargeting input and interfaces were PointRight and iCrafter. PointRight is a general pointer redirection system initially designed to allow a single mouse and keyboard to control all the displays and machines in an iRoom. It has been expanded into a general architecture that allows many different configurations and modes of use. For example, users running PointRight can use the pointer and keyboard from their laptops to point and type on the big displays in the iRoom [PR02]. ICrafter is a programmable UI builder that automatically formats controls for services in

the iRoom into buttons and boxes UI's, tailored to the size and capabilities of particular displays[iCrafter01]. Both PointRight and iCrafter help expand the desktop metaphor to a heterogeneous collection of linked machines.

iStuff is a project to explore experimental user interfaces beyond the desktop metaphor. We started by designing a variety of wireless buttons and sliders that were linked to the iRoom through the iROS infrastructure. Events from these devices could serve directly as input to applications, or as triggers for other events. For example, we can create a wireless "start the room" button whose event triggers a sequence of events to turn on the lights and start up the projectors in the iRoom.

The iStuff framework was designed to minimize the hardware design, construction and low-level programming that were not core parts of the research we wanted to conduct. This framework allows us to quickly prototype a non-standard physical user interface and run experiments with it, without running wires, soldering up components, and writing yet another serial device driver. In this way, iStuff is like Phidgets[Phidgets01], another toolkit for prototyping physical input and output devices.

IStuff differs from Phidgets in that we focus on the domain of interactive rooms, and assume the event-based infrastructure of the iROS to support the iStuff infrastructure. This allows iStuff to more easily be wireless and lightweight. All iRoom aware applications already include code to interface to the Event Heap, so it is trivial to link them to iStuff. The Event Heap and iStuff use a pure Java API, and are therefore available to Windows, Mac OS X, and Unix applications.

Because iStuff belongs to the iRoom, not to a particular machine or application, it forces us to address the issue of input that is not tied to a specific machine or display. Both the ability to quickly prototype physical devices, plus the flexible infrastructure that supports using them in exploratory applications, is helping us to investigate some fundamental questions about user interface software architectures in environments beyond the traditional desktop

Basic iStuff Architecture

We split the actual iStuff devices into two parts: one is the actual physical wireless device, and the other is a receiver and related software running on some PC that serves as a *proxy* in the room. Conceptually, both together make the iStuff "device," but this design makes the actual wireless end device lightweight, simple, and cheap to reproduce, since much of its "intelligence" is actually contained in the proxy (see Figure 2).

iStuff devices use a variety of technologies to do the actual wireless transmission, depending on what is most convenient. We have implemented garage-door-opener style Radio Frequency (RF) transmitters, the X-10 house automation

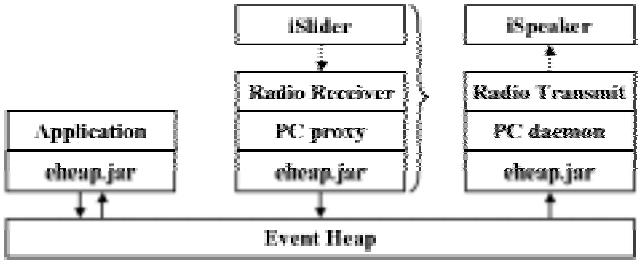

Figure 2. Basic iStuff architecture: Applications send and receive events through the Event Heap. An iSlider device is wirelessly connected to its proxy that converts its input into Event Heap events indicating steps along the slider. In the example for the output direction, the PC daemon receives Event Heap events for sound files to play, then plays them through its sound output where they are transmitted as analog radio frequency waves. The wireless iSpeaker is simply a portable radio.

protocol, infrared, analog radio, Bluetooth, and others. The point is that the technology does not matter to those who access the device in their software applications; it is hidden by the proxy. Using this simple architecture, we have implemented a variety of iStuff devices, as shown in figure 3.

We have also explored voice commands and audio feedback using iStuff. Voice is interpreted and translated to iStuff events, events containing text (to be translated to speech) or audio files are sent to speakers or a wireless radio in the iRoom.

Our device designs are freely available for reproduction at www.stanford.edu/~borchers/istuff/. Others can choose to use our hardware and software designs, create their own device designs, or purchase compatible commercial de-

Figure 3.Various iStuff. The iDog sends a button press event when turned over. The iPen is an augmented SMART board pen, where the embedded button operates as a "right click" to the Windows OS. The X10 buttons are standard X10 hardware. All other devices work through a simple RF receiver that plugs into the USB port of the Proxy PC.

vices, depending on their sense of adventure when it comes to electronics. The only thing required for new designs is the software for the proxy interfacing to the Event Heap.

IStuff with Intermediation

Sending and receiving events is only the tip of the iceberg when it comes to flexible interfaces for interactive rooms. How do we design the input architecture so that specific hardware is flexibly targeted to specific displays and applications? We certainly don't want to hardwire specific iStuff event types into the applications, as this would make them dependent on specific input devices. Our goal is to easily retarget input as needed.

One solution is to design one or more levels of intermediation that translate from one event type to another, and to provide simple tools for configuring this intermediation. Applications specify their input in their own terms (i.e. annotation event), input devices are built to produce generic, low-level events (i.e. iStuff #7 button press), and the intermediation system maps one to the other.

We are currently developing the *Patch Panel* as a general mechanism for this kind of mapping. The Patch Panel consists of a backend application that is responsible for mapping events of one type to another. The actual configuration of the Patch Panel backend is done using its graphical front end. It is designed to be accessible using a web browser, so any display in the iRoom can be turned into a controller to configure this mapping. In fact, the front end talks to the back end using the Event Heap, sending events of type *Patch Panel Configuration*.

For example, we have an application used for meeting capture that logs a variety of image and file data in the iRoom. Users can insert named annotations into the log by pressing iButtons. To do this, each person selects a button, then goes to a web page to configure it. The web page asks for the person's annotation (typically, the user's name), and asks them to press their iButton. It then captures the next iButton pressed event, and uses that information to create a configuration that maps that iStuff event to the annotation event for the meeting capture application. From then on, pressing the iButton enters the user's annotation into the log, which can then be used as a search string when the log is explored.

Towards the Great Unified Input Theory

The following questions are issues that our work on iStuff have highlighted. We do not have answers ready for them, but we think iStuff helps us to discover and think about these problems much better:

Input focus in the room: In a single-display, single-user desktop environment, it is obvious where the *focus* of the system is at any time (the active front-most window, typically selected via the window manager by click-to-type at the current mouse pointer position). It is unclear, however, what focus means in an interactive room: There are several screens, potentially being used in parallel, and multiple users, as well as multiple input devices. The same is true for the concept of *selection.* Focus and selection can be established by gaze, gesture, touch, voice, and other modalities; which modalities work best is one of the questions we are currently exploring, using iStuff for experimentation.

Multi-user, multi-devices: The ideal iRoom application can be controlled completely using iStuff -- it does not require a local input device, but simply can be configured using the Patch Panel to listen to any number of semantically compatible input devices. This requires rethinking of how operating systems and applications deal with events: In a room, an application has to be able to process input from *multiple users* using *multiple devices* in parallel on a single screen. Typically, this is impossible for three reasons: The hardware cannot distinguish, say, two mice (try plugging two USB mice into a computer); the operating system cannot deal with multiple cursors; and even if it could, applications are typically only written to deal with a single cursor and focus. We have begun to write applications such as a multi-screen, multi-machine whiteboard called the *iWall* that lets multiple users control objects on the same or different screens in parallel, using a variety of iStuff devices. It is interesting to see how natural this seems to users, and yet how far it is away from today's mainstream interactive software architectures, even though this domain has been the topic of HCI research since the early 90's [MMM92]

Feedback and latency: Our architecture ultimately suggests that all input and output events are passed over the local network. This creates problems with providing immediate feedback that local operating systems do not have, or are optimized to avoid: If the larger part of a second goes by between a user pressing an iButton and, say, the room turning on the lights in response, then users do not feel at ease and in control of the environment. Local feedback inside the wireless device is just part of the answer since it is not guaranteed that the wireless signal will actually get through to have the desired effect. Clearly, the robustness and failure tolerance of the iROS software infrastructure (and networks in general) is not designed for reliable immediate responses as needed in user interface development. Yet, ubicomp environments inherently depend on networked communication for by their distributed nature.

Multi-pointer input: The PointRight wireless mouse is logically iStuff, as it is a wireless device that supplies input to the iRoom. The original PointRight uses custom-built, direct socket connections to minimize latency for communicating pointer data. More recent implementations have demonstrated that pointer data can route through the Event Heap with adequate performance for 6-8 simultaneous users. This allows us to explore multi-pointer input via the iStuff architecture, such as multiple users using multiple pointing devices to move objects on the iWall. Besides stressing performance, pointer input requires efficient con-

trol of coordinate systems and unambiguous sequencing. Experiments with the iWall suggest our architecture will support this, but this has yet to be fully demonstrated.

Layered architecture: Our architecture is currently being generalized to implement a simple two-layer model of generalizing events. On the first layer, device-specific events (such as a certain interval of values a wireless slider can send) is generalized into a generic, normalized device value space (such as [0..1]). On the second layer, this normalized device value is being mapped to the application-specific data type and range (such as "March 2001" to "June 2002" for a time slider in a project scheduling application). Should there be more layers? Can one generically map many input devices to an application, or must they be mapped one-by-one? What about rich modalities like voice? It is easy to create an iStuff voice command that mimics a simple button press (and we have done it), less clear how to incorporate a more complex dialog.

Summary and Future Work

The iStuff framework provides a toolkit of lightweight, wireless, generic user interface components that can be easily reproduced, and controlled by simple software across multiple platforms. Its design has helped us explore basic questions about user interfaces in the post-desktop era of ubiquitous computing: Will there be universally adopted input modalities as in today's desktop world? How will users specify focus and select object in an augmented reality such as an interactive workspace? And how do software architectures need to change to deal with ubicomp scenarios and event flows? In the workshop, we hope to be able to share our experience with the iRoom, our enthusiasm about this new world of post-desktop user interfaces, and to learn about hypotheses and potential answers to these questions. We are well aware there is a wealth of related work, only some of which we have cited directly in this paper. Clearly "Searching For The Great Unified Input Theory," must build on the community's experience, and this workshop seems a great way to start building a common foundation on which to base this goal.

ACKNOWLEDGMENTS

Many students have worked on iStuff and the iRoom, especially: Josh Tyler, Merrie Ringel, Tico Ballagas, Michael Champlin, Robert Brydon, Jeff Raymakers, Joyce Ho and Ya'ir Aizenman. Terry Winograd is one of the original designer's of the iRoom, and has been a general source of wisdom for this work. This work was supported by DoE grant B504665, The Wallenberg Global Learning Network, and by donations of equipment and software from Intel Corp., InFocus, IBM Corp. and Microsoft Corp.

REFERENCES

[MMM] Bier, Eric A., Steve Freeman, Ken Pier, MMM: The multi-device multi-user multi-editor, *CHI92*

[iROS02] Johanson, Brad, Armando Fox, and Terry Winograd, The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing Magazine 1(2)*, April-June 2002.

[CIFE02] Fischer, Martin; Stone Maureen; Liston, Kathleen; Kunz, John; Singhal, Vibha (2002). "Multi-stakeholder collaboration: The CIFE iRoom." Proceedings CIB W78 Conference 2002: Distributing Knowledge in Building, Aarhus School of Architecture and Centre for Integrated Design, Aahus, Denmark, pp. 6-13.

[Mural01] Guimbretière, François, Maureen Stone, Terry Winograd, Fluid Interaction with High-resolution Wall-size Displays, *UIST 2001*.

[eHeap02] Johanson, B. and Fox, A., "The Event Heap: A Coordination Infrastructure for Interactive Workspaces," To appear in *Proc. of the 4th IEEE Workshop on Mobile Computer Systems and Applications (WMCSA-2002)*, Callicoon, New York, USA, June, 2002.

[PR02] Johanson, B., Hutchins, G., Winograd, T., Stone, M. PointRight: Experience with Flexible Input Redirection in Interactive Workspaces, *to be published at UIST 2002*.

[Myers98] Myers, Brad A., Herb Stiel, and Robert Gargiulo. "Collaboration Using Multiple PDAs Connected to a PC." Proceedings CSCW'98: *CSCW 1998*, pp. 285-294.

[Phidgets01] Saul Greenberg and Chester Fitchett. "Phidgets: Easy development of physical interfaces through physical widgets." In *Proceedings of the UIST 2001* pp 209–218

[Rekim99] Rekimoto, J. "Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments." *CHI'99*, pp. 378-385.

[iCrafter01] Ponnekanti, S., Lee, B., Fox, A., Hanrahan, P. and Winograd, T. "ICrafter: A Service Framework for Ubiquitous Computing Environments *Proc. Ubiquitous Computing Conference (UBICOMP), 2001*.

[Streitz99] Streitz, Norbert A., Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, Ralf Steinmetz, i-LAND: An interactive Landscape for Creativity and Innovation, *CHI99*

[Tandler01] Tandler, Peter: Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices. In: *Proceedings of UbiComp2001: Ubiquitous Computing*. Heidelberg: Springer LNCS 2201, 2001, pp. 96-115.