PROJECT SUMMARY

Making it Easier to Interact with Technology Through Handheld Personal Universal Controllers

We propose to investigate how various kinds of handheld devices can be used to control all kinds of home, office and factory equipment. Stereos, VCRs, telephones, copiers, FAX machines, factory equipment, clocks, and even light switches use computer-control interfaces. The problem with many appliances is that they are too complex. Some appliances need thirty or more buttons to cover all of their functions. This complexity can even make relatively simple tasks, like setting the clock on a VCR, so difficult that people avoid them. Even simple devices can be hard to use if the interface is unfamiliar to the user; when traveling, many people are stumped by the user interface for setting the alarm on the clocks in hotel rooms, so they arrange for a wake-up-call instead.

Most people today carry at least one form of wearable or handheld technology, including watches, cell-phones, pagers, or personal digital assistants (PDAs) like a Palm Pilot or PocketPC. In the near future, these devices will be able to communicate with each other, and with other appliances around them, using wireless technologies such as infrared, BlueTooth radio, or IEEE 802.11 radio.

Our proposal is that when users point their own handheld at a light switch, at a photocopier in an office, at a machine tool in a factory, at a VCR at home, at a piece of test equipment in the field, or at almost any other kind of device, the device will send to the handheld a description of its control parameters. The handheld uses this information to create an appropriate control panel, taking into account the properties of the controls that are needed, the properties of the handheld (the display type and input techniques available), and the properties of the user (what formats are familiar for various tasks, what language is preferred, how big the buttons should be based on whether the user prefers using a finger or a stylus). The user can then control the device using the handheld. The device will not need to dedicate much processing power, hardware, or cost to the user interface, since it will only need to contain a description of its capabilities and storage for the current settings, along with hardware for wireless communication. The handheld software will use intelligent techniques to create useful and appropriate interfaces that are customized for each user.

The approach we propose will help insure that the resulting user interfaces on the handheld will be easy to use. First, we are creating control panels by hand for various types of handhelds and various devices. We will then perform user studies to validate and improve these designs. A preliminary study suggests, for example, that hand-designed control panels for a shelf stereo and an office phone that runs on a Palm Pilot may be *twice as fast* with *1/5 the errors* as the manufacture's interface. The hand-designed user interfaces will guide the design of a specification language that will describe the components of the interface at a high level. The next phase of the research will be to create software that will automatically design interfaces from the specification language for different devices. Finally, we will perform user studies to determine the usability of the automatically generated interfaces, especially compared to the original hand-designed interfaces.

TABLE OF CONTENTS

Section		Total No. of Pages in Section	Page No.* (Optional)*
Cover Sheet (NSF Form 1207 - Submit Page 2 with original proposal only)			
A	Project Summary (NSF Form 1358) (not to exceed 1 page)	1	A.1
B	Table of Contents (NSF Form 1359)	1	B.1
C	Project Description (NSF Form 1360) (including Results From Prior NSF Support) (not to exceed 15 pages)	15	C.1

D	References Cited (NSF Form 1361)	5	D.1
E	Biographical Sketches (NSF Form 1362) (Not to exceed 2 pages each)	2	E.1
F	Summary Proposal Budget	7	F.1
	(NSF Form 1030, including up to 3 pages of budget justification/explanation)		
G	Current and Pending Support (NSF Form 1239)	2	G.1
H	Facilities, Equipment and Other Resources (NSF Form 1363)	1	H.1
Appendix	Letter of Certification; Human Subjects	1	

Making it Easier to Interact with Technology Through Handheld Personal Universal Controllers

1. Introduction

Increasingly, home and office appliances, including televisions, VCRs, stereo equipment, refrigerators, washing machines, thermostats, light switches, telephones, copiers, and factory equipment, have embedded computers, and often come with remote controls. Many predict that most future appliances will be on the Internet so they can be remotely monitored and controlled. However, the trend has been that as appliances get more features and are more computerized, their user interfaces get harder to use [Brouwer-Janse 1992].

Meanwhile, another trend is that people are increasingly carrying computerized devices that can communicate. People have cell-phones, pagers, personal digital assistants (PDAs) such as a Palm Pilot or PocketPC, and even watches that can communicate using various wireless networks. The advent of the BlueTooth short-distance radio network [Haartsen 1998] is expected to enable many devices to communicate with other devices in the vicinity.

Our proposed research is to try to use handheld devices to improve the user interfaces for appliances. Our preliminary research suggests that this has the opportunity to be spectacularly successful. For example, our prototype of control panels for a shelf stereo and an office telephone enabled users to complete complex tasks in 1/2 the time and with 1/5 the errors compared to using the manufacturer's interfaces [Nichols 2001]. Since many appliances are just now becoming networked and wireless capabilities are improving, this is the opportune time to be performing this research.

We call our approach the "Personal Universal Controller" (PUC). A key feature is that, unlike so-called "universal remotes" that are available today, such as the Phillips Pronto [Philips 2001], our proposed controller will be *self-programming*. Today's remotes are either pre-programmed in the factory with a subset of the features of some specific appliances, or else the user is required to laboriously hand-program the remote with each desired function of each device. In contrast, our proposed remote will engage in a two-way exchange with the appliance, to first upload a description of the appliance's functions from which it will create a control panel automatically, and finally to send appropriate control signals to the appliance as the user operates the control panel.

1.1. Complexity of Devices

An important motivation for the proposed research is the increasing complexity of consumer and business devices. Most desk phones, cell phones, clock radios, VCRs, stereos, washing machines, microwave ovens, thermostats, photocopiers, fax machines, etc. have many unused features. Most consumers find it difficult to master the basic functions of some of these devices, never mind the sophisticated features. Even simple devices are not immune to this problem; when traveling, I am frequently stumped by the user interface for setting the alarm on the clocks in hotel rooms.

One reason is that appliances must economize on buttons and displays, and therefore they often reuse the same buttons for multiple functions. Often, pressing and holding a button will perform a different operation than a quick tap, but there is usually no indication of this on the button's label. Many appliances have invisible temporal modes that change the meaning of buttons. Furthermore, most appliances do not provide unambiguous feedback to users. Indicators of appliance state can be confusing. For example, on a stereo that combines a CD and tape player, it may be difficult to decide whether a circling arrow means that the CD will repeat, the tape will repeat, or both. Feedback also must be given in response to operations so that the users know exactly what they are doing. As an example, it can be difficult to tell if a single beep is intended as positive or negative feedback.

In addition, there is little standardization of the labeling, placement or behavior of the controls on consumer devices. The increasing computerization of equipment can only make this problem worse, as it becomes cheaper to embed sophisticated computing in even the smallest devices. Today, enormous processing power can be very cheaply embedded in devices as simple as a light-switch, but providing a good user interface for that processor is still a significant expense. Many devices today come with custom remote controls, which themselves may be poorly designed. Furthermore, the consumer is faced with the familiar problem of having dozens of incompatible remote controls to deal with.

A CHI'92 panel states: "User interfaces for consumer products are notoriously bad" [Brouwer-Janse 1992] and claims that the causes include "designs that are modeled after the user interface of computer systems, … [and a] rigid system of constraints on display size, memory, cpu power, input devices, conditions of use, component price, mechanical compatibility, manufacturability and serviceability." Most devices probably do not get the extensive usability testing [Nielsen 1993] that might help identify usability problems early in the design phase. Hugo Strubbe from Philips says:

> Human factors techniques have been tried successfully on TV interface prototypes. However, such work typically has little effect on products. Designs made by human factors people are often more expensive than those made by engineers. Cost is an important purchase criterion for consumers. They rarely evaluate ease-of-use in the shop. Therefore, one cannot charge extra for it. Consumers who are unable to use the product at home accept this as their fault and do not return the product. The consumer has to be taught to insist on ease-of-use, and our human factors work does not directly contribute to this [Brouwer-Janse 1992, p. 289].

1.2. Target Platforms

In order to insure that our software is not specific to a particular form factor or operating system, we will target a number of different platforms for the PUC. Initially, we will focus on Palm OS devices, PocketPC devices, and Internet-enabled cell-phones. The Palm and PocketPC devices have monochrome or color screens that are touch sensitive. The PocketPC has a large screen with higher resolution. Cell-phones tend to have much smaller screens (e.g., 100x100 pixels) that are not touch sensitive. Instead, they use number buttons and scroll arrows.

In the future, we may look at supporting other devices, such as alphanumeric pagers with little keyboards and even a computerized watch [Narayanaswami 2000], if it becomes practical and available. In this document, we use the term "handheld" to refer to the platform hosting the Personal Universal Controller, but this is not meant to imply that we are excluding other kinds of wearable devices like watches for future consideration.

The devices to be controlled include all kinds of office and home equipment. Although we use the term "appliance" in this proposal, we also mean to include other equipment, such as light switches, copiers, automobile settings, factory equipment, and any other devices that may have embedded computers.

1.3. Advantages for a Personal Universal Controller

The proposed Universal Personal Controller aims to alleviate the problems with the user interface of today's appliances. Wouldn't it be nice to be able to point your own PDA at the hotel clock and use a familiar and well-designed user interface on the PDA to set the alarm? There are a number of reasons why our proposed Personal Universal Controller is expected to have a better user interface than today's devices.

By having a universal device owned by the person, the user interface will be separate from the functionality. This can allow appropriate specialists to be involved with the separate parts. Furthermore, by offloading the user interface onto a user's handheld, more money and effort can be spent on the handheld controller than would be practical for the devices being controlled.

Since the controller will be personal, the interfaces will be portable and consistent across appliances. The interfaces will belong to the user, not to the machine. Common user interface elements can be used across appliances, which will facilitate consistency and familiarity. If the user knows how to set the time for an alarm clock, the same user interface can be used to set the time on a VCR.

Another important advantage of the PUC is the ability to use more expensive hardware for the controller itself. For example, we may have a touch-sensitive graphical LCD screen such as found on PDAs and on high-end cell-phones, even though these are too expensive for use on conventional appliances.

As discussed below in section 5.2, our early research suggests that an interface on a handheld can be significantly better than the interface supplied by the manufacturer using the front panel of the appliance and its remote control.

1.4. Advantages for Automatic Design of Control Panels

The most interesting and challenging part of the proposed research is to *automatically* create panels of controls for appliances. Most remote controls today are pre-programmed in the factory for a single device, and sometimes for a few others. For example, most VCR remotes can also control TVs. This approach is clearly not scaleable to many devices, and the remotes often only cover the basic functions and omit useful features the consumer might want (e.g., mine does not have a "mute" button). The other approach used today requires the consumer to laboriously hand-program each function of the remote. For example, with the Phillips Pronto remote control [Philips 2001], the Harman-Kardon and Microsoft remote control [Harman-Kardon 2000], and Pacific Neo-Tek's "Omni-Remote" software [Pacific Neo-Tek 2000] for the Palm, the user must have the old remote for the device to be controlled. The old remote is aimed at the controller, and, for each operation to be provided on the controller, the user must specify a button on the controller and then push the corresponding button on the old remote. In contrast, the proposed PUC will be able to create a control panel for new devices without requiring any programming by the user.

One approach might be to have the device send to the PUC a fully-designed control panel. This is similar to the philosophy of Jini [Sun 2000], which sends a complete implementation of the user interface for the requested service. For the device control envisioned here, sending a pre-defined panel will not work because of the wide range of handheld mobile devices, with vastly different input and output capabilities. We want to support cell-phones with 8-line displays and 14 buttons, monochrome and color Palm OS devices with 160x160 pixels and a touch screen, color Pocket PC devices with 240 x 320 pixels, etc. In the Jini model, the device would need to provide many different control panels each tuned for a different kind of handheld. Furthermore, the devices we want to control are expected to last for many years whereas the features of handhelds change every year. For example, we would want the same light switch or dishwasher that can be controlled by a PDA to work with a future controller built into a watch, which might have a new kind of round display and only four buttons.

Therefore, we propose to have the appliance send to the controller a high-level specification of the parameters and controls for the device, and have the controller automatically create a user interface based on this specification. This will enable the same specification to be used by controllers with significantly different characteristics. The automatic generation of the user interface will take into account the input and output capabilities of the controller along with user preferences and other information. Users might specify, for example, that they want bigger buttons and bigger labels because they want to press them with a finger, or alternatively that they prefer smaller buttons so more will fit on the screen at the same time. Since the display on the handheld is likely to be bigger than on the device, longer names and even help text can appear on the controller. If automatic machine language translation is available, the user might be able to request a French or Japanese interface, and have the textual labels translated by the controller.

Another important advantage of automatic generation is that the controller can impose consistency across all the devices that are controlled. For example, both a phone and a clock radio have a volume control, so panels for both should use the same widget for volume and put it in the same place. For example, a controller on a Pocket PC will use the standard widget for setting the volume (a slider next to a speaker icon), but on a Palm, the volume usually is represented as a pop-up with a few choices. Similarly, when the controller is pointed at a clock from any manufacturer, the same control panel should appear on the controller. The panels should also be consistent with other applications running on the same hardware.

Dynamically created user interfaces to networked devices, such as telephones, enable new kinds of services to be automatically incorporated into the user interface even after the devices have been deployed. For example, if the phone system starts offering "call-back-last-number," this might automatically appear as a new labeled button on the PUC, rather than requiring the user to remember to press *69.

Another possible advantage is that the controller can serve as an "authentication token" and dynamically provide different interfaces to different classes of users. Some devices, such as the 3Com Internet phones [Dalgic 2000], require the user to login. Pointing a PDA at the phone is much easier than keying in a user name and password onto the phone keypad. If stronger authentication is needed, technologies like Secure Computing's SafeWord or SofToken (http://www.securecomputing.com/), which already run on Palm OS and Pocket PC, might be incorporated into the controller protocols. Once the user is authenticated to the device, then new capabilities could be enabled. For example, the device might only provide certain controls to certain classes of users. For example, televisions in hotels might only provide the controls to add new channels to authorized service people, whereas anyone's controller would be able to change channels and volume. Thermostats in offices might allow anyone to change the temperature within certain ranges, but only certain people might be authorized to switch from cooling to heating.

1.5. Smart Environments

Many research groups (e.g., CMU's Aura project: http://www.cs.cmu.edu/~aura/; Stanford's Interactive Workspaces project: http://graphics.stanford.edu/projects/iwork/: and MIT's Oxygen project http://www.oxygen.lcs.mit.edu/) are investigating rooms that will have embedded cameras and microphones to enable the user to gesture and talk to control the environment. When people carry a wireless mobile data communication device into these so-called "smart environments," the device should serve as another way to control the environment. For example, sometimes it may be appropriate to speak a command (e.g., saying "lights on"), but other times, it may make more sense to use a control panel on a handheld mobile device (e.g., to move a continuous slider to adjust the light brightness, rather than saying "darker, darker, darker, OK" or "lights to 32 percent").

The infrastructure that is needed for appliance control by smart environments is similar as what is needed by a Personal Universal Controller as proposed here. Both cases will need to be able to discover what devices are around, get descriptions of what the devices can do, and know how to give commands to control the devices. Therefore, we see the PUC research as very complementary to the Smart Environments work.

1.6. Controlling Real Appliances

The exciting vision of the Personal Universal Controller proposed by this research can clearly not happen until all appliance manufacturers adopt a consistent standard for communicating with the PUC, and on the specification language for the devices. It is not expected that this would happen during this research project (if it ever does). However, the goal of this project is to investigate the format and content of such a specification and algorithms that might be used to generate the interfaces. We hope that this research will help motivate the manufacturers to adopt such a standard, and we would be willing to help with the standardization effort.

In the short term, there will clearly not be devices that can communicate with our Personal Universal Controller. We will be able to test and evaluate our research progress using simulations of the specifications that the appliances would return, and then simulations of the control of the appliances. However, it will be much more interesting and exciting to be able to demonstrate the actual control of some devices. We will first acquire whatever devices we can find that support two-way inquiry and control. For example, 3Com Corporation project has created a Palm remote controller for an Internet phone [Dalgic 2000], which we hope to acquire for demonstration purposes. We are also negotiating with companies such as Sony and Philips to try to acquire devices with which we can engage in two-directional communication because we very much want to demonstrate practical and effective control of some real devices. We will also select a few devices that already accept infrared one-way remote control, and create specifications for what they would report if they supported two-way communication. These specifications will then be used to generate user interfaces on the controller as if they had been supplied directly by the devices. We plan to investigate using a "universal remote" like the Philips Pronto, the Harman-Kardon remote, or Pacific Neo-Tek's "Omni-Remote" software as a front end to control devices these kinds of devices.

2. Related Work

There are many different classes of systems that are related to the proposed work.

A number of research groups are working on controlling appliances from handheld devices. Hodes, *et. al.* propose a similar idea to our PUC, which they call a "universal interactor" that can adapt itself to control many devices [Hodes 1997]. However, their research seems to have focused on the system and infrastructure issues rather than how to create the user interfaces. Their later paper describes the "rvic" system [Hodes 1999] that allows a Palm pilot or laptop to remotely control the audio/video equipment in a meeting room, but the control panels are hand-designed and hard-coded into the Palm program. The Stanford iRoom project [Fox 2000] also supports remote control from PDAs, and they tried two designs: one with the remote control hand-coded on the Palm, and the other using Web forms displayed by a standard Web browser on the handheld. In both cases, the programmer designed the control panels in advance. The IBM PIMA project mentions using a PDA to control devices and services [Banavar 2000], but apparently has not yet addressed this issue. Another IBM project [Eustice 1999] describes a "Universal Information Appliance" (UIA) that might be implemented on a PDA. The UIA uses an XML-based language called MoDAL from which it creates a user interface panel for accessing information. However, the MoDAL processor apparently only handles simple layouts and its only type of input control is text strings.

A part of the Xweb [Olsen Jr. 2000] project is working to create technologies that can create customized interfaces that are appropriate to the interests of the user. The goal is to separate the functionality of the appliance from the device upon which it is displayed. Xweb defines an XML language from which user interfaces can be created. Another XML language for user interface design is UIML [Abrams 1999], from which user interfaces can be created.

Other projects have looked at the general issues around having a PDA and stationary devices working together, including the original Xerox ParcTab [Want 1995] system, Rekimoto's many systems

[Rekimoto 1997][Rekimoto 1998][Rekimoto 1999], and our Pebbles system [Myers 1998b][Myers 2000b][Myers 2000c][Myers 2001b]. In these, the user interfaces for the PDA have been hand-designed.

With respect to automatic design of user interfaces, the WML language for WAP phones is relevant, since it leaves some aspects of the user interface for the phone to decide. However in practice, most of the design must be included in the WML specification. There were a number of research systems that looked at automatic design of user interfaces for conventional computers. These sometimes went under the name of “model-based” techniques [Szekely 1993]. Here, the programmer provides a specification (“model”) of the properties of the application, along with specifications of the user and the display. This approach was moderately successful at creating dialog boxes [Kim 1993] [Vander Zanden 1990] and creating complete interfaces in a limited range [Olsen Jr. 1989] [Frank 1993] [Szekely 1993]. The ITS system from IBM was used to create all the screens for the information kiosks at the EXPO’92 worlds fair [Wiecha 1989][Wiecha 1990]. Of particular note is the layout algorithm in the DON system that achieved a pleasing, compact, and logical placement of the controls [Kim 1993]. Other systems focused on the initial creation assuming a user would edit the resulting user interface [Foley 1988] [Singh 1989]. We plan to extend these results to create panels of controls on handhelds of significantly different properties.

One problem with many of these model-based systems is that the automatically created user interface was often not as good as a person could create, and therefore required intervention to fix up the resulting interface or else fix up the rules with special cases. Another problem is that the specification the designer had to write got to be quite large with much extra information needed for the layout algorithm to do a good job. We feel that our proposed system will avoid these problems because we will start off with high-quality user interfaces from which the specification language features will be designed, and we can use the restrictions of the control panels to simplify the specifications. Controller interfaces are simpler than other kinds of interfaces since they are constructed mainly of buttons and selection lists, grouped appropriately on the screen. Understanding the groupings solves much of the layout problem, which is key issue that will be addressed in the specification language.

3. The Pebbles Project

The research proposed here will be performed as part of the Pebbles project (http://www.pebbles.hcii.cmu.edu) [Myers 2001b], which is investigating the many ways that handheld devices will be used at the same time as other computerized devices. As part of the Pebbles project, we have looked at multiple PDAs connected to a PC to support meetings [Myers 1998b]. For example, in design reviews, brainstorming sessions, and organizational meetings, a PC is often used to display slides or a current plan, and the people in attendance provide input. Our applications allow each person to use their PDA to control the PC’s cursor and keyboard input from their seat. In other research, we are investigating how a PDA can be equally useful for a single person as an extension to the Windows user interface for desktop applications [Myers 2000c]. We performed a study that showed, for example, that the PDA could be used very effectively as a scrolling device for desktop applications [Myers 2000b].

The software we have developed as part of the Pebbles project is mostly available for free download from our web site: http://www.pebbles.hcii.cmu.edu. We anticipate that the technology developed as part of the proposed research will similarly be widely distributed.

4. Device Discovery and Communication

How will the Personal Universal Controller find out about the devices? There is already a move towards “self-describing devices.” The BlueTooth radio standard [Haartsen 1998], for example, already includes a “Service Discovery Protocol” for finding out whether there are any other Bluetooth-enabled devices in a given space.

Many other “service discovery” and “service description” languages are being developed, usually based on the XML format. Currently, most of these are aimed at peripherals that today connect to computers, but extensions to other kinds of devices are inevitable. For example, Microsoft’s “Universal Plug and Play” (http://www.upnp.org) includes a way for devices to describe their features, and has been embraced by “more than 120 consumer electronics, computing, home automation, home security, home appliance, computer networking and other leading companies” (quoted from a press release at: http://www.upnp.org/forum/default.htm). Other efforts include the Jini protocol [Sun 2000], Salutation (http://www.salutation.org/), parts of the WML standard for cell-phones [WAP Forum 1999], and UIML that is a proposed language for describing user interfaces [Abrams 1999]. Another example is that 3Com has developed an Internet telephone that can interface to a Palm handheld [Dalgic 2000]. The Palm can control certain features of the telephone using a standard protocol called “PhoneControl” [Dean 2000] that 3Com has proposed as an IETF standard. PhoneControl provides for 2-way communication and allows the controller to inquire as to the capabilities of the telephone, as well as to initiate and terminate calls. Unfortunately, none of these standards seem sufficiently powerful for what we need in a specification language. It is interesting to note that the ability to query about devices’ properties and to control devices remotely will also be necessary to enable other kinds of external control that many people are predicting, including centralized computer control and voice control of devices.

Communicating with devices will also get increasingly easy. Today, many systems already have embedded infrared receivers, used by their remote controls. In the future, devices may support two-way infrared communication to enable the controller to receive information from the device as well. The 3Com Internet Phone works this way [Dalgic 2000]. All of today’s PDAs and many cell-phones also have two-way infrared capability (even the Furby toy has two-way infrared). The advantage of infrared is that it is already available and cheap. Infrared is usually highly directional and short range, which is both an advantage and a disadvantage. It minimizes security concerns if only controllers near a device can control it (people on the street cannot turn on your TV), and people can identify which device they want to control by aiming the controller at it. Disadvantages include that it is not possible to find out all the devices that are around, and the user may not know where to point the device (to control the lights, do you point the controller at the light itself, or do you need to point to the “regular” wall switch for that light?).

Other wireless technologies, such as IEEE 802.11, digital cell-phones and two-way pagers may also be useful. The new BlueTooth technology is advertised to provide short-range (in a room) low-power omnidirectional radio communication. Projections are that the chips to support BlueTooth will eventually match the cost and power usage of today’s infrared chips. Many vendors, from cell-phones to PDAs, are expected to incorporate BlueTooth communication. BlueTooth promises to significantly expand the range of devices that can be remotely controlled. It is inherently a two-way communication mechanism. Security is built into the BlueTooth specification, since it is designed for use with consumer telephones and other personal devices. We are hopeful that BlueTooth will be useful for finding out what devices are nearby and for communicating with devices without requiring the user to know where to point the controller.

Currently, we envision using a combination of BlueTooth and infrared technologies in our Personal Universal Controller research. We hope to get early BlueTooth devices when they become available. Initially, we expect to use plug-in modules such as the Widcomm BlueTooth card for the HandSpring Visor (http://www.widcomm.com/products/blueConnect.htm). For infrared communication, we can use technology such as Omni-Remote [Pacific Neo-Tek 2000] to control the built-in infrared port on a Palm PDA.

5. Automatic Design of Control Panels

Automatically creating high quality control panels from a specification is the main focus of this proposal. It is a difficult research problem that is unlikely to be addressed by commercial vendors, and has been mostly omitted from other work on remote controllers. Some previous approaches to automatic user interface generation only produced low-quality, rough outlines of the user interface, and required much work from the user to improve the interface [Foley 1988] [Singh 1989]. Our design task will be made more difficult by the desire to support handhelds such as cell-phones with impoverished input and output capabilities. Nielsen warns that usability is increasingly important as the device gets smaller and has fewer resources [Nielsen 2000].

In order to automatically create high-quality user interfaces, we propose a multi-phase approach. First, we will hand-design control panels for a variety of devices and handhelds, and then evaluate these designs with users. This phase is already in progress. Next, we will study these panels to determine the important properties that seem to be necessary in order for the system to be able to generate similar interfaces. This will drive the design of a specification language that will be able to represent all of the important properties. Next, automatic generation engines will be created for various platforms that will be able to generate interfaces. The resulting interfaces will be evaluated in user tests for a variety of appliances to compare the generated interfaces to the hand-drawn ones, and to the original appliances' interfaces.

5.1. Hand-drawn screens

As an initial step, we hand-designed control panels for a variety of appliances (see Figure 1). We did a Palm user interface for a complex office telephone and answering machine (the AT&T 1825), for a clock-radio (the Bose Wave Radio), and for a shelf stereo with its remote control (the Aiwa CX-NMT70). We also did part of the user interface for the clock-radio for a Windows CE screen, and part of the telephone interface on an Internet phone.

It is important to note that the complexity of these screens is much greater than can be handled by the remote controllers in the related work, and the specifications that can describe them seem to be beyond what can be represented in MoDAL, UIML, etc. We need this level of complexity, however, in order to cover the full set of functions of today's (and tomorrow's) appliances.

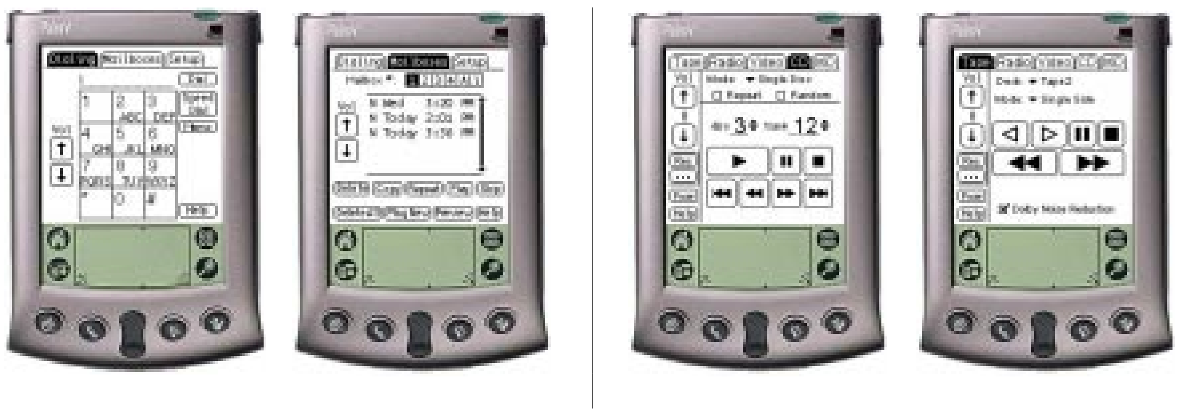

Figure 1. Hand-drawn control panels for the Palm for the AT&T phone (left) and AIWA stereo (right).

5.2. Initial User Study

We then performed a user study to see if we were on the right track. We compared the manufacture's interface to a paper-prototype of our Palm interfaces for the AT&T telephone and the Aiwa stereo. Figure 1 shows a few screens used in the study. We chose these two appliances because both are common and combine several functions into a single unit. The Palm interfaces cover all the features of these appliances, as described in their user manuals. We used paper prototypes [Rettig 1994] because a software implementation has not yet been completed. The results suggest that our on-screen control panels are much easier to use than the original physical buttons on the devices, especially for complex tasks.

Subjects were asked to work through one list of tasks for the stereo and another list for the phone. One set of subjects worked on the actual stereo and the prototype interface for the phone, while the others worked on the actual phone and prototype for the stereo. We recorded the number of missteps and the number of times external help was required while the tasks were performed.

We anticipated that some subjects would not be able to complete some of the more difficult tasks. If a subject gave up while working with the actual phone or stereo, they were given the user manual and asked to complete the task. Subjects working on the paper prototype were required to press the “help” button, available on nearly every screen, to get a verbal hint of how to proceed.

Thirteen Carnegie Mellon graduate students, five female and eight male, volunteered to participate as subjects. All subjects were enrolled in the School of Computer Science. All had significant computer experience. Seven owned Palm devices at the time of the study, one subject had no Palm experience, and the remaining five had exposure to Palm devices in class or through friends. Everyone in the group had some experience with stereo systems. Only two did not have a stereo. Four subjects happened to own a stereo of the same brand used in this study.

The results of the study indicate that subjects made fewer missteps and asked for help less using the prototype handheld interfaces than using the actual appliances (see Figure 2). All of the results are statistically significant at the $p < 0.001$ level using a **what kind of test??**. On average, the Palm user interface subjects made about 1/5 the errors (missteps). This indicates that the prototype handheld interfaces were more intuitive to use than the actual interfaces. Further, the average time to complete the tasks was about twice as long with the real appliance interface as with the Palm prototypes.

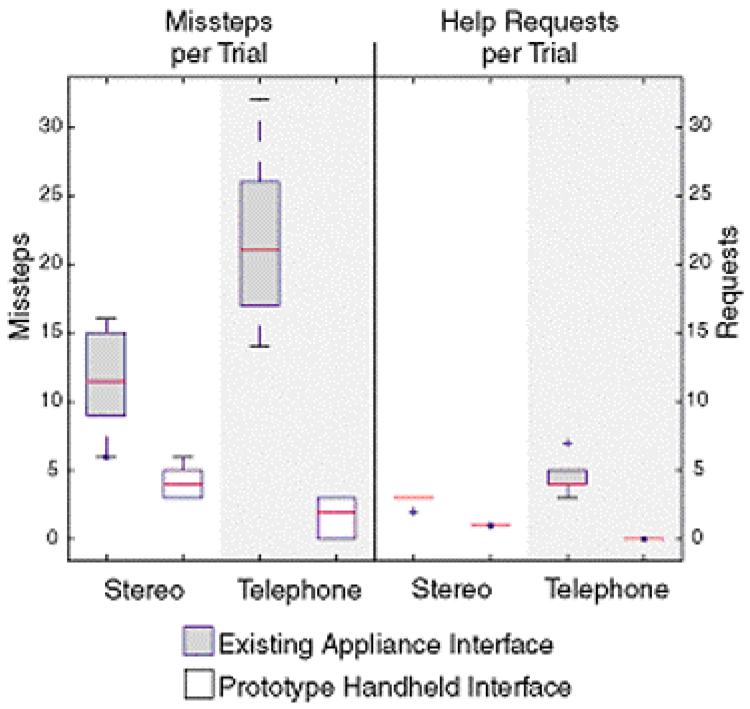

Figure 2. Box-and-whisker plots showing the range of missteps and help requests for each appliance and interface type.

The problems that users had with the existing interfaces were mainly due to poorly labeled buttons and inadequate feedback. The worst examples of poorly labeled buttons were found on the AT&T phone. This phone has several buttons that can be pressed and released to activate one function and be pressed and held to activate another function. There is no text on the appliance to indicate this.

A similar problem is also encountered on the stereo. Setting the timer requires the user to press a combination of buttons, each button press within four seconds of the last. The stereo does not display an indicator to warn of this restriction, and often users were confused when a prompt would disappear when they had not acted quickly enough.

The phone also suffered from an underlying technical separation between the telephone and the answering machine. None of the buttons on the phone can be used with the answering machine. Even the numeric codes for the answering machine must be set using arrow buttons rather than the keypad. All but one subject tried to use the keypad buttons to set the code. The one exception read the user manual.

In general, the prototype handheld interfaces were more intuitive. Informal measurements of time indicate that subjects needed half as much time to complete tasks on the prototype interface. This difference can be accounted for in part by the large number of times subjects needed help with a task using the existing interface versus the paper prototype. Subjects almost never asked for help using the paper prototype.

Inspired by the significant success of this preliminary study, we feel it is very appropriate to move ahead with the proposed research. In parallel with hand-designing interfaces for other platforms (such as the WAP internet phone and the PocketPC) and other appliances (such as a copier and FAX machine), we will begin designing the specification language and automatic generation tools.

5.3. Specification Language

We have some preliminary ideas about the requirements and the design for the specification language that will be used to create the user interfaces. The most important requirement is that it must be sufficiently flexible and powerful so that an automatic generator can create high-quality user interfaces. The specification must be able to cover all the features of all appliances, so the PUC can be used to do both simple and complex tasks. In addition, we would like it to be easy to write for the human designer who is specifying an appliance's features. Ideally, the size of a specification for an appliance should be smaller than the size of specifying one completely designed user interface.

Another goal is that the specification should be at a high-level, describing the functions and operations of the appliance, rather than how they are realized in any user interface. Only in this way can the software on the handheld have the flexibility to create customized interfaces. The next sections discuss our initial thoughts on some details for the specification language.

5.3.1. High-Level Types

Most work on automatic generation of user interfaces has concentrated on primitive types, such as strings, commands, numbers, and enumerated types (select one or more from a list). The goal of the software was then to assign appropriate widgets to each (text input fields, buttons, sliders, and radio buttons, check boxes, or menus respectively), and then lay out the widgets. In the proposed research, we will extend the primitive types with higher-level types, such as "Volume," for which some PUCs may have special widgets. For example, a Pocket PC implementation should use the standard built-in widget for setting the volume (a slider next to a speaker icon). We will also investigate high-level types that are appropriate for our set of appliances. For example, specifying a time is a common activity across many appliances. Another common element might be a phone number, which often would be entered with a standard telephone-like pad. Many appliances support dynamic lists, such as the list of tracks on a CD, or the messages and pre-sets on a phone. To allow for future expansion, we will not require that each handheld have an implementation for every higher-level type. Instead, each higher-level type will be able to be decomposed into one or more primitive types. For example, volume might be a number between 1 and 10.

The proposed system will also be able to add extra information to the types to help the user interface generator. For example, lists will be annotated with the usual number of entries to help choose the appropriate layout. String values will have formatting information to help the controller display them appropriately, and for input fields, to determine if the user has entered a legal value.

As an example, a specification for the parameter “Power” on a radio might include the following information (the syntax and values of this example are just meant to be illustrative):

```
Type: Power         // high-level type for which there may be a special widget
Is-A: Boolean       // primitive type, in case there isn’t a widget for power: value can be true or false
Label: Power || {Radio}[Off,On]     // various choices for how this might be labeled
Importance: High    // to tell the layout handler to make this element more prominent
```

5.3.2. The Structure of the Interface

The most interesting part of the specification language will be the *relationships* of the properties and controls to each other. Many appliances are highly moded, and various controls are only visible in certain modes. For example, for the stereo, whole groups of controls depend on which source for audio is selected (CD, tape, radio, etc.). On a control panel for a phone, the “hang-up” button should only be available when the phone is in use. Depending on the controller, the hang-up button might either be grayed out or else it might be invisible when it is not available. For other kinds of controls, the default or current value will be dynamically determined depending on other settings. These dependencies will need to be in the specification that the device sends to the controller.

Another kind of relationship that seems to be needed is the grouping of related objects. For example, on a tape player, the buttons for play, stop, pause, fast-forward, rewind, and record should be together. We investigated grouping in our earlier work [Vander Zanden 1990] and similar techniques may be used for the PUC.

The groupings cannot be inferred from the names of the controls. For example, both a telephone and a TV have a mute button. On the TV, mute is associated with volume, since it sets the volume to zero, so the mute button makes sense to group with volume. However, on a speakerphone, the volume sets the loudness that you hear, whereas mute makes it so the person at the other end cannot hear you. Therefore, for the phone, the mute button has nothing to do with the volume control. Although both functions are labeled “mute,” good controller designs would probably lay them out differently.

At a higher-level, there will be groupings that might be put on different panes on small screens, or put in pop-up dialog boxes. This relates to the “card” concept in WML for WAP [WAP Forum 1999]. For example, on the phone, the widgets for dialing might be put on a different pane than the widgets for voice mail messages. This grouping will have to be flexibly described so the interface can be adapted to both small screens where there may be many panes, and to large screens where many controls may fit on the screen at once.

5.3.3. Representation of the Specification Language

The above requirements for the specification language will help guide the selection of the particular syntax and representation. Our current thoughts are to use some form of XML as the representation, since it is broadly used and available. It is easy for people to read and there are many tools to help write and parse XML. We are also investigating various XML-based languages, such as uPnP, WML, MoDAL, UIMS, and PhoneControl, to see if they can be adapted to serve our needs. Clearly it would be better to adopt some standard, but as a research project, we feel we should first determine the requirements for generating a high-quality user interface, and then evaluate the different options to see which ones are best.

Another interesting issue with the specification language is that it is for two-way communication, not just for describing the appliance's properties in one batch file. Ideally, the handheld can control the appliance by sending back data in the same format as used for the specification. For example, if the specification includes that there is a parameter of type "Power" whose current value is "Off," the controller should be able to send back a similar message saying the value has changed to be "On."

Another use for two-way communication will be when the controller needs further elaboration for the user interface itself. In many cases, it may be appropriate to limit the initial communication to only the essential information, to save time and battery power for the transfer. When further information is needed, the controller might ask for it from the appliance. This may happen when the controller does not have a higher-level type, and then it may ask the appliance to send the primitives. Another case might be if the user clicks on a "help" button on the controller, the appliance may send over context-dependent help. Further, if the controller has a color screen and high-bandwidth communication, it may ask for fancy icons from the appliance rather than using standard textual labels.

5.4. Automatic Design

In parallel with developing the specification language, we will also be creating software for generating user interfaces for the panels from the specifications. In our previous work on automatic creation of user interfaces [Vander Zanden 1990], we showed that we could generate reasonable quality user interfaces that varied based on the look-and-feel of the underlying platform. This work and that of others (e.g., [Kim 1993]), augmented by more rules to insure that the layout conforms to the best design practices, will form part of the basis of the proposed implementation. We hope to be able to use a machine-independent programming language for the implementation, such as Java, so that we do not need to re-implement the software for the various machines.

An important consideration for the algorithm will be to achieve consistency across different appliances. This goes beyond just using consistent widgets and terminology. It might also encompass trying to enforce similar behaviors on sets of widgets. For example, it applies to layouts since the software should try to place the volume control at about the same place for control panels of different appliances. It might also apply to entire groups of controls. For example, if the user is used to setting a clock using hour and minute buttons, a new appliance's time should be set using that familiar set of controls, and not, for example, using "fast" and "slow" buttons. Enabling this consistency is another good reason that it is better to dynamically create the control panels rather than having the appliance pre-specify what they will look like, since the appliance will not know about the control panels for other devices that the user has seen.

The controller interfaces are expected to be simpler to generate than other kinds of interfaces. The interfaces are constructed mainly of buttons and selection lists, grouped appropriately on the screen. Understanding the groupings solves much of the layout problem, which is problem that we will address in the specification language. We also will use mode knowledge from the specification to isolate elements that control modes from those elements that are change depending on the mode.

The goal will be that the generated user interfaces should be very close to the hand-designed panels we started with. To measure whether they are "close enough," user studies will be performed to compare the hand-designed and automatically generated panels against the original physical devices. The automatically generated interfaces should be able to match the performance of the hand-designed panels.

5.5. User Customizations

After the controller has automatically generated a user interface, we propose to allow the user to customize it in various ways. This will use technology like in our Shortcutter application [Myers 2000c], which already provides the ability to hand-create and hand-edit panels of controls. We expect that only advanced users will want to customized their interfaces, but it does provide many advantages to those who do:

- Elimination of buttons for functions that are never used, to simplify the interface. Alternatively, the user might make the buttons for common functions be more prominent.
- Supporting default values for frequently used parameters. For example, a clock controller can always come up ready to set the alarm to 7:00am.
- Constructing control panels that control *multiple* devices, by dragging in controls from different panels. Macros can even be created so that a single button will send multiple commands to different devices. For example, a button might lower the lights while turning on the TV, the VCR and the cable box.
- Sharing customizations. Third parties might even provide a market of specially designed control panels. For example, for the Philips Pronto, there is a web site where one can get many different pre-programmed panels that people have created [Philips 2001].

These customizations would be saved for the user and used for all devices of the same type. Techniques that combine designer-specified and user-specified constraints [Borning 1997] might be used to implement the customizations.

6. Other Benefits

If the proposed research is successful, it will have benefits beyond just remote control devices for appliances. The proposed research will help further the cause of separating the user interface from the application code, which has always been a basic goal of user interface software research from the beginning. Using a high-level specification of the application functionality to guide the automatic design of the user interface may be useful for many other kinds of applications, besides remote controllers. As future mobile and handheld devices have increasingly diverse user interface characteristics, and yet can run the same software (for example, if it is written in a portable language such as Java), it will be increasingly important for the user interface to automatically adapt to whatever platform it is on. The techniques developed as part of the proposed research may point the way for how to create a more general user interface generator for many portable computerized applications.

7. Conclusions

The proposed research addresses important research questions in the development of “smart products” for home automation and in the use of handheld devices with wireless communication systems. The Personal Universal Controller would make an exciting commercial product that many people would probably like to have. However, significant research is necessary before such a controller can be created, especially to investigate what form a specification language should take to describe the functions, and how high-quality user interfaces can be created from that specification that are adapted to the characteristics of the user and the handheld. Devices are already increasingly able to communicate and accept external control. If we can show success in this research, then manufacturers may be more motivated to also have the devices export the specifications of their parameters.

8. Results from Brad Myers’s Prior NSF Grants

Brad Myers has two current NSF grants, and had two NSF grants that have completed.

One current grant is IRI-9900452, “A More Natural Programming Environment for Children,” which is partially supporting the PhD work of John Pane. This project is investigating how to make programming easier to children by investigating more natural ways to expressing programming concepts, and by using good user interface design practices in the design process. The background studies have been reported in a journal article [Pane 2001], three conference papers [Pane 2000c][Pane 2000a][Pane 2000b] and two keynote addresses [Myers 1999][Myers 2000a]. Many further publications and results are expected.

The other current grant is IIS-9817527 supported by the Digital Library Initiative-2, and is for “An Intelligent Authoring Tool for Non-Programmers Using the Informedia Video Library.” This project, with Drs. Scott Stevens and Al Corbett, is investigating how to create a very easy-to-use editor for digital video. This is supporting the MS work of one student and the PhD work of another, and has resulted in a CHI’2001 student poster [Casares 2001] and a conference paper in submission [Myers 2001a]. This research is just getting started.

Brad Myers’s two prior NSF grants supported work on a number of projects in the general area of *demonstrational interfaces.* A demonstrational interface extends the range of direct manipulation by allowing the user to operate on examples from which the system creates a generalization. The fundamental goal of these projects is to give to users the *capabilities* of programming without requiring them to learn how to program. The first grant, IRI-9020089, called “Using Demonstration in User Interfaces,” from 8/15/91 until 1/31/94, supported research on demonstrational interfaces for *text formatting styles*, and for creating scripts in a *visual shell*. The work on text formatting, which was the Master’s thesis for Andrew Werth [Werth 1992], allowed the user to select a region of text containing different parts which are formatted differently, and the system tried to figure out from the example which formatting goes with which parts of the document. A *visual shell* is an iconic interface to a file system (e.g., the Apple Macintosh Finder). The PURSUIT system, which was the Ph.D. thesis of Francesmary Modugno [Modugno 1995], allows the user to start recording, perform a sequence of commands, and then stop recording. The primary innovations in PURSUIT are the automatic generalization of the program so it will be more reusable, and the novel representation of the program so the user can verify and edit the program. PURSUIT represents the recorded program as a “comic strip” which uses familiar file and folder icons, and shows the changes to those icons using before and after pictures representing the operations [Modugno 1997b].

The second grant, IRI-9319969, called “Demonstrational Interfaces for Visualization and End-User Programming,” from 8/15/94 until 9/30/97, funded research into demonstrational interfaces for visualization (charting) as well as an architecture to support creating demonstrational interfaces. In the Gold system [Myers 1994], the user can quickly draw a small piece of the picture of what the desired chart will look like and the system will generalize to create a chart appropriate to the picture and the selected data. This resulted in a patent and the BS thesis of Andrew Faulring (1999). The architecture, which was the Ph.D. work of David Kosbie, explored the recording of actions at multiple levels. These *hierarchical events* [Kosbie 1993] allow the user and automatic mechanisms to choose which level of the recording is appropriate. These ideas were also used to develop the *hierarchical command objects* [Myers 1996] that are part of the Amulet user interface development environment.

In summary, the publications directly resulting from Myers’s four NSF grants include:

- 1 Ph.D. thesis by Francesmary Modugno [Modugno 1995] (with 2 PhDs in progress),
- 1 Masters thesis by Andrew Werth [Werth 1992] (with 1 in progress),
- 1 BS thesis by Andrew Faulring (1999),
- 1 patent: “Creating Charts and Visualizations by Demonstration,” Patent Number 5,581,677, April 22, 1994.
- 7 book chapters [Kosbie 1993][Modugno 1993b][Myers 1993b][Myers 1993a][Myers 1993c][Myers 2001c][Wolber 2001],

- 5 refereed journal articles [Myers 1992][Modugno 1997a][Modugno 1997b][Myers 2000d][Pane 2001],
- 18 refereed conference publications [Myers 1991a][Myers 1991b][Myers 1991c][Modugno 1993a][Myers 1993d][Werth 1993][Kosbie 1994][Modugno 1994a][Modugno 1994c][Modugno 1994b][Myers 1994][Modugno 1996][Myers 1996][Myers 1998a][Pane 2000c][Pane 2000a][Pane 2000b]
- 1 student poster [Casares 2001]
- 2 conference keynote addresses [Myers 1999][Myers 2000a].

References

[Abrams 1999] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams and Jonathan E. Shuster. "UIML: An Appliance-Independent XML User Interface Language," *The Eighth International World Wide Web Conference,* Toronto, Canada, May 11-14, 1999.http://www8.org/ and http://www.uiml.org/

[Banavar 2000] Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman and Deborra Zukowski. "Challenges: An Application Model for Pervasive Computing," *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000),* 2000.http://www.research.ibm.com/PIMA/

[Borning 1997] Alan Borning, Richard Lin and Kim Marriott. "Constraints and the Web," *Proceedings of the 1997 ACM Multimedia Conference,* 1997. pp. 173-182. http://www.cs.washington.edu/research/constraints/web/mm97.html

[Brouwer-Janse 1992] Maddy D. Brouwer-Janse, Raymond W. Bennett, Takaya Endo, Floris L. van Nes, Hugo J. Strubbe and Donald R. Gentner. "Interfaces for consumer products: "how to camouflage the computer?"" *CHI'1992: Human factors in computing systems,* Monterey, CA, May 3 - 7, 1992. pp. 287-290.

[Casares 2001] Juan P. Casares. "SILVER: An Intelligent Video Editor," *ACM CHI'2001 Student Posters,* Seattle, WA, March 31-April 5, 2001. p. To appear. http://www.cs.cmu.edu/~silver/CasaresShortPaper.pdf

[Dalgic 2000] Ismail Dalgic, Michael Borella, Rick Dean, Jacek Grabiec, Jerry Mahler, Guido Schuster and Ikhlaq Sidhu. *True Number Portability and Advanced Call Screening in a SIP-Based IP Telephony System*. 3Com Corporation. http://www.3com.com/technology/tech_net/white_papers/503054.html. March 14, 2000.

[Dean 2000] Rick Dean and Ronnen Belkind. *PHONECTL Protocol; Internet Engineering Task Force (IETF) draft standard*. 3Com, Rolling Meadows, IL. http://www.cs.columbia.edu/%7Ehgs/sip/drafts/draft-dean-phonectl-01.txt. Sept., 2000.

[Eustice 1999] K.F. Eustice, T. J. Lehman, A. Morales, M. C. Munson, S. Edlund and M. Guillen. "A Universal Information Appliance," *IBM Systems Journal.* 1999. **38**(4). pp. 575-601. http://www.research.ibm.com/journal/sj/384/eustice.html

[Foley 1988] James D. Foley, Christina Gibbs, Won Chul Kim and Srdjan Kovacevic. "A Knowledge-Based User Interface Management System," *Human Factors in Computing Systems,* Proceedings SIGCHI'88. Washington, D.C., May, 1988, 1988. pp. 67-72.

[Fox 2000] Armando Fox, Brad Johanson, Pat Hanrahan and Terry Winograd. "Integrating information appliances into an interactive workspace," *IEEE Computer Graphics and Applications.* 2000. **20**(3). pp. 54-65.

[Frank 1993] Martin R. Frank and James D. Foley. "Model-Based User Interface Design by Example and by Interview," *ACM SIGGRAPH Symposium on User Interface Software and Technology,* Proceedings UIST'93. Atlanta, GA, Nov, 1993, 1993. pp. 129-137.

[Haartsen 1998]Jaap Haartsen, Mahmoud Naghshineh, Jon Inouye, Olaf J. Joeressen and Warren Allen. "Bluetooth: Vision, Goals, and Architecture," *ACM Mobile Computing and Communications Review.* 1998. **2**(4). pp. 38-45. Oct. www.bluetooth.com.

[Harman-Kardon 2000] Harman-Kardon. *Take Control System Controller*. Washington, D.C., http://www.harmankardon.com/products/productlines/takecontrol.asp. 2000.

[Hodes 1997] Todd D. Hodes, Randy H. Katz, Edouard Servan-Schreiber and Lawrence Rowe. "Composable ad-hoc mobile services for universal interaction," *Proceedings of the Third annual*

ACM/IEEE international Conference on Mobile computing and networking (ACM Mobicom'97), Budapest Hungary, September 26 - 30, 1997. pp. 1 - 12.

[Hodes 1999] Todd Hodes, Mark Newman, Steven McCanne, Randy H. Katz and James Landay. "Shared Remote Control of a Video Conferencing Application: Motivation, Design, and Implementation," *Proceedings of SPIE Multimedia Computing and Networking*, San Jose, CA, January, 1999. pp. 17-28. http://daedalus.cs.berkeley.edu/publications/mmcn99.ps.gz

[Kim 1993] Won Chul Kim and James D. Foley. "Providing High-level Control and Expert Assistance in the User Interface Presentation Design," *Human Factors in Computing Systems,* Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993. pp. 430-437.

[Kosbie 1993] David S. Kosbie and Brad A. Myers. "A System-Wide Macro Facility Based on Aggregate Events: A Proposal," *Watch What I Do: Programming by Demonstration,* Cambridge, MA, MIT Press. 1993. pp. 433-444.

[Kosbie 1994] David S. Kosbie and Brad A. Myers. "Extending Programming By Demonstration With Hierarchical Event Histories," *Human-Computer Interaction: 4th International Conference EWHCI'94, Lecture Notes in Computer Science, Vol. 876,,* Berlin, Springer-Verlag. 1994. pp. 128-139.

[Modugno 1993a] Francesmary Modugno. "Pursuit: Programming in the Interface," *Adjunct Proceedings INTERCHI'93,* Presentation to the Doctoral Consortium. Amsterdam, The Netherlands, Apr, 1993a. p. 217.

[Modugno 1995] Francesmary Modugno. *Extending End-User Programming in a Visual Shell with Programming by Demonstration and Graphical Language Techniques.* Pittsburgh, PA, Computer Science Department, Carnegie Mellon University. 1995. PhD Thesis. Computer Science Technical Report CMU-CS-95-130.

[Modugno 1996] Francesmary Modugno, Albert T. Corbett and Brad A. Myers. "Evaluating Program Representation in a Demonstrational Visual Shell," *Empirical Studies of Programmers: Sixth Workshop,* Ablex Publishing Corporation. 1996. pp. 131-146.

[Modugno 1997a] Francesmary Modugno, Albert T. Corbett and Brad A. Myers. "Graphical Representation of Programs in a Demonstrational Visual Shell -- An Empirical Evaluation," *ACM Transactions on Computer-Human Interaction.* 1997a. **4**(3). pp. 276-308.

[Modugno 1994a] Francesmary Modugno, T.R.G. Green and Brad A. Myers. "Visual Programming in a Visual Domain: A Case Study of Cognitive Dimension," *Proceedings of Human-Computer Interaction '94, People and Computers IX,* Glasgow, Scotland, 1994a. pp. 91-108.

[Modugno 1993b] Francesmary Modugno and Brad A. Myers. "Graphical Representation and Feedback in a PBD System," *Watch What I Do: Programming by Demonstration,* Cambridge, MA, MIT Press. 1993b. pp. 423-431.

[Modugno 1994b] Francesmary Modugno and Brad A. Myers. "Pursuit: A Demonstrational Visual Shell," *Technical Video Program of the CHI'94 Conference,* SIGGRAPH Video Review, Issue 89, No. 17. 1994b.

[Modugno 1994c] Francesmary Modugno and Brad A. Myers. "A State-Based Visual Language for a Demonstrational Visual Shell," *1994 IEEE Workshop on Visual Languages,* IEEE Computer Society. St. Louis, MO, Oct, 1994, 1994c. pp. 304-311.

[Modugno 1997b] Francesmary Modugno and Brad A. Myers. "Visual Programming in a Visual Shell -- A Unified Approach," *Journal of Visual Languages and Computing.* 1997b. **8**(5/6). pp. 276-308.

[Myers 1999] Brad Myers. "Towards More Natural Domain-Specific Languages," *DSL'99: 2nd Usenix Conference on Domain Specific Languages,* Invited Keynote Speaker. Austin, TX, October 3-6, 1999.

[Myers 2000a] Brad Myers. “Creating More Natural Programming Languages,” *VL’2000: IEEE Symposium on Visual Languages,* (Invited Keynote Address). Seattle, Washington, September 10-14, 2000a.http://www.cs.orst.edu/~burnett/vl2000

[Myers 1996] Brad A Myers and David Kosbie. “Reusable Hierarchical Command Objects,” *Proceedings CHI’96: Human Factors in Computing Systems,* Vancouver, BC, Canada, April 14-18, 1996. pp. 260-267.

[Myers 1991a] Brad A. Myers. “Demonstrational Interfaces: A Step Beyond Direct Manipulation,” *People and Computer VI,* Cambridge, Cambridge University Press. 1991a. pp. 11-30.

[Myers 1991b] Brad A. Myers. “Text Formatting by Demonstration,” *Human Factors in Computing Systems,* Proceedings SIGCHI'91. N.O., LA, Apr, 1991, 1991b. pp. 251-256.

[Myers 1992] Brad A. Myers. “Demonstrational Interfaces: A Step Beyond Direct Manipulation,” *IEEE Computer*. 1992. **25**(8). pp. 61-73.

[Myers 1993a] Brad A. Myers. “Demonstrational Interfaces: A Step Beyond Direct Manipulation,” *Watch What I Do: Programming by Demonstration,* Cambridge, MA, MIT Press. 1993a. pp. 485-512.

[Myers 1993b] Brad A. Myers. “Tourmaline: Text Formatting by Demonstration,” *Watch What I Do: Programming by Demonstration,* Cambridge, MA, MIT Press. 1993b. pp. 309-321.

[Myers 1998a] Brad A. Myers. “Scripting Graphical Applications by Demonstration,” *Human Factors in Computing Systems,* Proceedings SIGCHI'98. Los Angeles, CA, Apr, 1998a. pp. 534-541.

[Myers 2001a] Brad A. Myers, Juan P. Casares, Scott Stevens, Laura Dabbish, Dan Yocum and Albert Corbett. “A Multi-View Intelligent Editor for Digital Video Libraries,” Submitted for Publication. 2001a.http://www.cs.cmu.edu/~silver/dl2001paper.pdf

[Myers 1991c] Brad A. Myers, Allen Cypher, David Maulsby, David C. Smith and Ben Shneiderman. “Demonstrational Interfaces: Coming Soon?,” *Human Factors in Computing Systems,* Proceedings SIGCHI'91. New Orleans, LA, Apr, 1991, 1991c. pp. 393-396.

[Myers 1994] Brad A. Myers, Jade Goldstein and Matthew A. Goldberg. “Creating Charts by Demonstration,” *Human Factors in Computing Systems,* Proceedings SIGCHI'94. Boston, MA, Apr, 1994. pp. 106-111.

[Myers 2000b] Brad A. Myers, Kin Pou ("Leo") Lie and Bo-Chieh ("Jerry") Yang. “Two-Handed Input Using a PDA And a Mouse,” *Human Factors in Computing Systems,* Proceedings CHI'2000. The Hague, The Netherlands, Apr 1-6, 2000b. pp. 41-48.

[Myers 1993c] Brad A. Myers and David Maulsby. “Glossary,” *Watch What I Do: Programming by Demonstration,* Cambridge, MA, MIT Press. 1993c. pp. 593-603.

[Myers 2001b] Brad A. Myers, Robert C. Miller, Benjamin Bostwick, Franklin Chen, Carl Evankovich, Herb Stiel, Karen Cross, Adrienne Warmack, Chun-Kwok Lee, YuShan Chuang, Marsha Tjandra, Monchu Chen, Kin Pou Lie, Bo-chieh Yang and Zack Rosen. “Using Hand-Held Devices and PCs Together,” *ACM Communications of the ACM.* 2001b. p. To appear.

[Myers 2000c] Brad A. Myers, Robert C. Miller, Benjamin Bostwick and Carl Evankovich. “Extending the Windows Desktop Interface With Connected Handheld Computers,” *4th USENIX Windows Systems Symposium,* Seattle, WA, August 3-4, 2000c. pp. 79-88.

[Myers 1998b] Brad A. Myers, Herb Stiel and Robert Gargiulo. “Collaboration Using Multiple PDAs Connected to a PC,” *Proceedings CSCW’98: ACM Conference on Computer-Supported Cooperative Work,* Seattle, WA, November 14-18, 1998b. pp. 285-294. http://www.cs.cmu.edu/~pebbles

[Myers 1993d] Brad A. Myers, Richard Wolf, Kathy Potosnak and Chris Graham. “Heuristics in Real User Interfaces,” *Human Factors in Computing Systems*, Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993d. pp. 304-307.

[Myers 2001c] Brad Myers and Richard McDaniel. “Demonstrational Interfaces: Sometimes You Need a Little Intelligence; Sometimes You Need a Lot.” *Your Wish is My Command.* H. Lieberman, Ed. 2001c: Morgan Kaufmann. p. To appear.

[Myers 2000d] Brad Myers, Richard McDaniel and David Wolber. “Intelligence in Demonstrational Interfaces,” *Communications of the ACM*. 2000d. **43**(3). pp. 82-89.

[Narayanaswami 2000] Chandra Narayanaswami and M.T. Raghunath. “Application Design for a Smart Watch with a High Resolution Display,” *Proceedings of the Fourth International Symposium on Wearable Computers (ISWC’00),* Atlanta, Georgia, 18 - 21 October, 2000. pp. 7-14. http://www.research.ibm.com/WearableComputing/factsheet.html

[Nichols 2001] Jeffrey W. Nichols. “Using Handhelds as Controls for Everyday Appliances: A Paper Prototype Study,” *ACM CHI’2001 Student Posters,* Seattle, WA, March 31-April 5, 2001. p. To appear.

[Nielsen 1993] Jakob Nielsen. *Usability Engineering*. Boston, Academic Press. 1993.

[Nielsen 2000] Jakob Nielsen. *Alertbox: WAP Field Study Findings*. Fremont, CA, Nielsen Norman Group. December 10, 2000. http://www.useit.com/alertbox/20001210.html

[Olsen Jr. 1989]Dan R. Olsen Jr. “A Programming Language Basis for User Interface Management,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 171-176.

[Olsen Jr. 2000]Dan R. Olsen Jr., Sean Jefferies, Travis Nielsen, William Moyes and Paul Fredrickson. “Cross-modal Interaction using Xweb,” *Proceedings UIST’00: ACM SIGGRAPH Symposium on User Interface Software and Technology,* San Diego, CA, 2000. pp. 191-200.

[Pacific Neo-Tek 2000] Pacific Neo-Tek. *OmniRemote*. http://www.pacificneotek.com. 2000.

[Pane 2000a] J. F. Pane and B.A. Myers. “Improving User Performance on Boolean Queries,” *CHI 2000 Extended Abstracts: Conference on Human Factors in Computing Systems,* The Hague, Netherlands, ACM Press. April 1-6, 2000a. pp. 269-270. http://www.cs.cmu.edu/~pane/CHI2000.html

[Pane 2000b] J.F. Pane and B.A. Myers. “The Influence of the Psychology of Programming on a Language Design: Project Status Report,” *Proceedings of the 12th Annual Meeting of the Psychology of Programmers Interest Group,* Corigliano Calabro, Italy, April 10-13, 2000b. pp. 193-205. http://www.cs.cmu.edu/~pane/PoPInfluence.html

[Pane 2000c] J.F. Pane and B.A. Myers. “Tabular and Textual Methods for Selecting Objects from a Group,” *Proceedings of VL 2000: IEEE International Symposium on Visual Languages,* Seattle, WA, IEEE Computer Society. September 10-13, 2000c. pp. 157-164. http://www.cs.cmu.edu/~pane/VL2000.html

[Pane 2001] John F. Pane, Chotirat A. Ratanamahatana and Brad A. Myers. “Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems,” *International Journal of Human-Computer Studies.* 2001. **54**(2). pp. 237-264. http://www.cs.cmu.edu/~pane/IJHCS.html

[Philips 2001] Philips. *Pronto Intelligent Remote Control*. Philips Consumer Electronics. 2001. http://www.pronto.philips.com/

[Rekimoto 1997] Jun Rekimoto. “Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments,” *ACM SIGGRAPH Symposium on User Interface Software and Technology,* Proceedings UIST'97. Banff, Alberta, Canada, Oct 14-17, 1997. pp. 31-39.

[Rekimoto 1998] Jun Rekimoto. “A Multiple Device Approach for Supporting Whiteboard-based Interactions,” *Human Factors in Computing Systems*, Proceedings SIGCHI'98. Los Angeles, CA, Apr, 1998. pp. 344-351.

[Rekimoto 1999] Jun Rekimoto and Masanori Saitoh. “Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments,” *Human Factors in Computing Systems,* Proceedings SIGCHI'99. Pittsburgh, PA, May, 1999. pp. 378-385.

[Rettig 1994] Marc Rettig. “Prototyping for Tiny Fingers,” *Comm. ACM*. 1994. **37**(4). pp. 21-27.

[Singh 1989] Gurminder Singh and Mark Green. “A High-Level User Interface Management System,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 133-138.

[Sun 2000] Sun. *Jini Connection Technology*. Sun Microsystems. http://www.sun.com/jini/. 2000.

[Szekely 1993] Pedro Szekely, Ping Luo and Robert Neches. “Beyond Interface Builders: Model-Based Interface Tools,” *Human Factors in Computing Systems,* Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993. pp. 383-390.

[Vander Zanden 1990] Brad Vander Zanden and Brad A. Myers. “Automatic, Look-and-Feel Independent Dialog Creation for Graphical User Interfaces,” *Human Factors in Computing Systems*, Proceedings SIGCHI'90. Seattle, WA, Apr, 1990. pp. 27-34.

[Want 1995] Roy Want, Bill N. Schilit, Norman Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis and Mark Weiser. “An Overview of the ParcTab Ubiquitous Computing Experiment,” *IEEE Personal Communications.* 1995. pp. 28-43. December. Also appears as Xerox PARC Technical Report CSL-95-1, March, 1995.

[WAP Forum 1999] WAP Forum. *Wireless Application Protocol Wireless Markup Language Specification: WAP WML, Version 1.2*. Wireless Application Protocol Forum, Ltd. http://www.wapforum.org/. November, 1999.

[Werth 1992] Andrew J. Werth. *Tourmaline: Formatting Document Headings by Example*. Information Networking Institute, Carnegie Mellon University, 1992.

[Werth 1993] Andrew J. Werth and Brad A. Myers. “Tourmaline: Macrostyles by Example,” *Human Factors in Computing Systems,* Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993. p. 532.

[Wiecha 1989] Charles Wiecha, William Bennet, Stephen Boies and John Gould. “Generating user interfaces to highly interactive applications,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 277-282.

[Wiecha 1990] Charles Wiecha, William Bennett, Stephen Boies, John Gould and Sharon Greene. “ITS: A Tool for Rapidly Developing Interactive Applications,” *ACM Transactions on Information Systems.* 1990. **8**(3). pp. 204-236.

[Wolber 2001] David Wolber and Brad Myers. “Stimulus-Response PBD: Demonstrating When as Well as What.” *Your Wish is My Command.* H. Lieberman, Ed. 2001: Morgan Kaufmann. p. To appear.

BIOGRAPHICAL SKETCH

Brad Myers

Brad A. Myers is a Senior Research Scientist in the Human-Computer Interaction Institute in the School of Computer Science at Carnegie Mellon University, where he is the principal investigator for various research projects including: Silver Multi-Media Authoring, Natural Programming, the Pebbles Handheld Computer Project, User Interface Software, and Demonstrational Interfaces. He is the author or editor of over 200 publications, including the books *Creating User Interfaces by Demonstration* and *Languages for Developing User Interfaces*, and he is on the editorial board of five journals. He has been a consultant on user interface design and implementation to about 40 companies, and regularly teaches courses on user interface software. Myers received a PhD in computer science at the University of Toronto where he developed the Peridot UIMS. He received the MS and BSc degrees from the Massachusetts Institute of Technology during which time he was a research intern at Xerox PARC. From 1980 until 1983, he worked at PERQ Systems Corporation. His research interests include handheld computers, user interface development systems, user interfaces, programming by example, programming languages for kids, visual programming, interaction techniques, window management, and programming environments. He belongs to SIGCHI, ACM, IEEE Computer Society, IEEE, and Computer Professionals for Social Responsibility.

Mailing Address:
Brad A. Myers
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
412-268-5150
FAX: 412-268-1266
bam@cs.cmu.edu
http://www.cs.cmu.edu/~bam

Five Publications Related to This Proposal

Brad Vander Zanden and Brad A. Myers. “Automatic, Look-and-Feel Independent Dialog Creation for Graphical User Interfaces,” *Human Factors in Computing Systems*, Proceedings SIGCHI'90. Seattle, WA, Apr, 1990. pp. 27-34.

Brad A. Myers. “Using Multiple Devices Simultaneously for Display and Control.” *IEEE Personal Communications*, Special issue on "Smart Spaces and Environments." vol. 7, no. 5, Oct. 2000. pp. 62-65.

Brad A. Myers, Robert C. Miller, Benjamin Bostwick, Franklin Chen, Carl Evankovich, Herb Stiel, Karen Cross, Adrienne Warmack, Chun-Kwok Lee, YuShan Chuang, Marsha Tjandra, Monchu Chen, Kin Pou Lie, Bo-chieh Yang and Zack Rosen. “Using Handheld Devices and PCs Together,” *Communications of the ACM*. Accepted for publication. To appear.

Brad A. Myers. “User Interface Software Tools,” *ACM Transactions on Computer-Human Interaction*. vol. 2, no. 1, March, 1995. pp. 64-103.

Brad A. Myers, *et. al*., “The Amulet Environment: New Models for Effective User Interface Software Development,” *IEEE Transactions on Software Engineering*, vol. 23, no. 6, June, 1997, pp. 347-365.

Five Other Publications

Brad A. Myers, ed. *Languages for Developing User Interfaces*. Boston: Jones and Bartlett, 1992.

Brad A. Myers. *Creating User Interfaces by Demonstration*. Boston, MA: Academic Press, May 1988.

Brad Myers, Scott E. Hudson, and Randy Pausch, "Past, Present and Future of User Interface Software Tools," *ACM Transactions on Computer Human Interaction*. March, 2000. vol. 7, no. 1. pp. 3-28.

Brad A. Myers, Herb Stiel, and Robert Gargiulo. "Collaboration Using Multiple PDAs Connected to a PC," *Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work*, November 14-18, 1998, Seattle, WA. pp. 285-294.

Brad Myers, Kin Pou ("Leo") Lie and Bo-Chieh ("Jerry") Yang, "Two-Handed Input Using a PDA and a Mouse," *Proceedings CHI'2000: Human Factors in Computing Systems*. April 1-6, 2000. The Hague, The Netherlands. pp. 41-48.

Collaborators

I have had close collaborations with Brad Vander Zanden, University of Tennessee, and Pedro Szekely, USC/Information Sciences Institute.

Students and Visitors

Ph.D. students (3 graduated, 4 in progress, 1 withdrew):

- **Graduated**: Francesmary Modugno (1995, now at Univ. of Pittsburgh), James Landay (1996, now Asst. Prof. at Univ. Calif. at Berkeley), Rich McDaniel (1999, now at Siemens Technology to Business Center, Berkeley, CA)
- **Withdrew**: David Kosbie (now at Microsoft)
- **In Progress**: Rob Miller, John Pane, Jeff Nichols, Laura Dabbish

Masters students (2 graduated, 1 in progress):

- **Graduated**: Andrew Werth, (1992), Nobuhisa Yoda (1994)
- **In Progress**: Juan Casares

BS thesis student (3 graduated):

- **Graduated**: Rajan Parthasarathy (1994), Chotirat Ratanamahatana (1998), Andrew Faulring (1999)

Ph.D. Thesis Committees (4 at CMU, 5 external)

- **At CMU**: Tom Lane (1990), Dean Rubine (1991), Michael Gleicher (1994), Nick Thompson (in progress)
- **External member, Thesis committee**: Bernhard Suhm (PhD, 1998, University of Karlsruhe, Germany), T. Paul McCartney (1996, Washington Univ. in St. Louis), Martin R. Frank (1995, Georgia Institute of Technology), David Maulsby (1994, University of Calgary), David Kurlander (1993, Columbia University).

Visitors Supervised (6)

- Philippe Marchal (1987-1988), Brad Vander Zanden, (Postdoc, 1988-1990), Osamu Hashimoto, (NEC, 1990-1991), Keiji Kojima (Hitachi, 1990-1991), Alex Klimovitski, (1994-1995), Prof. Yoshihiro Tsujino (Osaka Univ, 1996-1997)

My Graduate Advisors

Master's advisor at MIT: David Reed. Ph.D. advisors at University of Toronto: William Buxton and Ronald Baecker.

FACILITIES, EQUIPMENT & OTHER RESOURCES

FACILITIES:

Laboratory:
The Human Computer Interaction Institute has available three usability labs that can be used for the evaluations as part of this research, although some of the evaluations will be performed in the field and in various meeting rooms. Our group has a computer laboratory containing a number of advanced workstations and personal computers that can be used for this project.

Clinical: Not applicable

Animal: Not applicable

Computer:
The proposed system will run primarily on various handheld platforms. Our group already has a collection of these computers, using PCs for development. The proposed budget includes funds for one more PC for each researcher and various PDAs. We may also need to purchase appliances that can be remotely controlled.

Heterogeneous Distributed Computing – The SCS research facility provides numerous and diverse computers for faculty and graduate-student use – more than 3000 machines. All have transparent access to the Andrew File System, a multi-terabyte, shared file space, and to one another through the Network File System protocol. SCS maintains several terabytes of secondary storage. Beyond these resources, the University provides various independent facilities for general use. Computationally intensive applications can also use PSC computers, including Cray T3E, C90-16/512, and J90 supercomputers.

Experimental Systems – SCS has a reputation for developing innovative computers, devices, networks, and systems that benefit diverse applications. Current large-scale, experimental efforts include the Darwin "application-aware" networking project and the NASD project on storage interfaces with direct device/client communication.

Networking – Carnegie Mellon operates a fully-interconnected, multimedia, multiprotocol campus network. The system incorporates state-of-the-art commercial technology and spans over 100 segments in a "collapsed backbone" infrastructure that enables mutual access among all campus systems, including the PSC supercomputers. The University currently provides 2Mb/s wireless data communication campus-wide.

Externally, SCS connects directly to the Internet, through OC-3 (155 Mbit/s) links, the NSF-sponsored vBNS (OC12) network, and the Internet-2 Abilene network. Carnegie Mellon is also actively engaged in the very high bandwidth research NGI initiatives.

Office: CMU provides office space for the investigators and their students and staff.

Other:
Carnegie Mellon's School of Computer Science (SCS) is the largest academic organization devoted to the study of computers. Its five degree-granting departments -- the Computer Science Department, Robotics Institute, Human-Computer Interaction Institute, Center for Automated Learning & Discovery, and Language Technologies Institute -- include over 200 faculty, 300 graduate students, and a 200-member professional technical staff. Two new units, the Entertainment Technology Center and the International Software Research Institute, opened in 1998 and 1999. SCS also collaborates with other University Research Centers, including the DoD-funded Software Engineering Institute (SEI); the NSF-sponsored Pittsburgh Supercomputing Center (PSC), the Information Networking Institute, and the Institute for Complex Engineered Systems (ICES).

MAJOR EQUIPMENT: None

OTHER RESOURCES: None