Studying The Use of Handhelds To Control Everyday Appliances

Jeffrey Nichols and Brad A. Myers
Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{jeffreyn, bam}@cs.cmu.edu

ABSTRACT

Everyday appliances, including telephones, ovens, and home stereos, increasingly contain embedded computers to provide greater functionality. Unfortunately, as these appliances become more complex, their interfaces are becoming harder to use. At the same time, more people than ever are carrying computerized devices that can communicate, such as cellular telephones, personal digital assistants, and even some watches. Our vision is that these devices will be able to communicate with our everyday appliances using a short-range wireless network, enabling people to control their appliances from a single handheld device. We present two studies that suggest that handheld devices could be used effectively as remote controls for everyday appliances.

Keywords: Handheld computers, remote control, appliances, Personal Digital Assistants (PDAs), Palm, PocketPC, Pebbles

INTRODUCTION

Increasingly, home and office appliances, including televisions, VCRs, stereo equipment, ovens, thermostats, light switches, telephones, and factory equipment, have embedded computers, and often come with remote controls. However, the trend has been that as appliances get more computerized with more features, their user interfaces get harder to use [3].

Meanwhile, another trend is that people are increasingly carrying computerized devices that can communicate. People have cellular phones, pagers, personal digital assistants (PDAs) such as the Palm Pilot or PocketPC, and even watches [16] that can communicate using various wireless methods. The advent of the BlueTooth short-distance radio network [8] is expected to enable many devices to communicate with other devices that are within close range.

Submitted for Publication

We are investigating how handheld devices can be used to improve the interfaces for home and office appliances, using an approach that we call the "Personal Universal Controller" (PUC). The concept of a PUC is similar to the universal remote controls that are available today for controlling many different consumer electronic products, such as the Philips Pronto [21]. Unlike the Pronto, which must be hand-programmed by a user, a PUC will be self-programming. This means that the PUC will engage in a two-way exchange with the appliance, first downloading a description of the appliance's functions, then creating a control panel automatically, and finally sending appropriate control signals to the appliance as the user operates the control panel.

The most interesting and challenging part of building a PUC is the *automatic* creation of control panels. In order to automatically create high-quality interfaces, we are taking a multi-step approach. First, we hand-design control panels for a variety of appliances and handhelds, and then evaluate these designs with users. Next, we will extract from these panels the important properties that that our system will need to use to generate similar interfaces. This will drive the generation of a specification language that will be able to represent the functional capabilities of an appliance. Finally, we will create automatic generation engines that accept this specification language. The first two phases of this project are described in this paper.

We have created hand-designs of remote control interfaces for a common stereo and a telephone/answering machine, and conducted two studies comparing them to the interfaces of the actual stereo and phone. The first study, described previously [17], compared paper prototypes [25] to the actual interfaces. The second study was similar to the first, except that actual implementations of our hand-designed interfaces were used instead of paper prototypes. Both studies showed that subjects were able to complete complex tasks using our hand-designed interfaces in about one-half the time and with half as many errors compared to using the actual interfaces.

This paper first describes our concept of a Personal Universal Controller in greater detail. We discuss our approach to building a PUC, including the first step of creating of hand-designed interfaces. We then describe the two studies of these interfaces, their results, and the application of their results to our long-term goals. We conclude by discussing our future work and the previous work that this project is based upon.

COMPLEXITY OF APPLIANCES

An important motivation for our investigation of the PUC is the increasing complexity of consumer and business appliances. Most appliances have many unused features, because many consumers find it difficult to master the basic functions of some of these appliances, never mind the sophisticated features. Even simple appliances are not immune to this problem; when traveling, I am frequently stumped by the user interface for the setting the alarm on the clocks in hotel rooms.

One reason that appliance interfaces may be difficult to use is that engineers must economize on buttons and displays, and thus reuse buttons for multiple functions. Often, pressing and holding a button will perform a different operation than a quick tap, but usually there is no indication of this on the button's label. Most appliances also economize the feedback given to the user. For example, an action might be confirmed with two beeps instead of a pre-recorded audio message. Visual indicators can also be ambiguous, like 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.

In addition, there is little standardization of the labeling, placement, or behavior of the controls on consumer appliances. The increasing computerization of equipment can only make this problem worse, as it becomes cheaper to embed sophisticated computing in even the smallest appliances. 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 [3]. Because of this, most appliances probably do not get the extensive usability testing [18] that might help identify usability problems early in the design phase.

UNIVERSAL CONTROLLERS

The Personal Universal Controller aims to alleviate the problems with the user interface of today's appliances. There are a number of reasons why we expect the PUC to have a better user interface.

A universal device separates the user interface from the functionality. By off-loading the interface onto a user's handheld, more money and effort can be spent on the handheld controller than would be practical for the appliances being controlled. These extra resources could be spent to improve the quality of the software and provide more extensive hardware on the handheld controller. For example, we may have a touch-sensitive graphical LCD screen such as those found on PDAs and high-end cellular phones, even though these are too expensive for use on conventional appliances.

There are at least three ways to build a universal controller that would have the properties described above:

- **Manually Programmed**, where either the factory, the user, or both must manually program the codes necessary to control a new appliance. The universal controllers on the market today, such as the Philips Pronto [21], are of this type.
- **Downloaded Interface**, where each appliance is shipped with a set of different user interfaces. The controller would download the interface built for its form factor from the appliance. This is similar to the philosophy of Sun Microsystems' JINI technology [27].
- **Self-Programmed**, where each appliance is shipped with a specification language that defines its functions. The controller would download this specification from the appliance and build an interface that takes into account the form-factor of the controller and preferences of the user.

Manually programmed controllers are available today, but are very inconvenient for their users because they must be updated every time a new appliance is purchased. The downloaded interface controllers solve this problem, as long as one of the interfaces stored in the appliance will work on the controller that the user has. Unfortunately, this is not a very scaleable approach, and is not suited for our purposes because we wish to support a wide-range of handheld mobile devices, such as cellular phones with 8-line displays, monochrome and color Palm OS devices with 160x160 pixel screens, color PocketPC devices with 240x320 pixels, etc. Another problem is that each type of handheld may have its own conventions for how interactions should be performed, and the downloaded interface would not be able to take these into account. Furthermore, the appliances we want to control are expected to last for many years, whereas the features of handhelds change rapidly. 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 round display and four buttons.

A self-programmed controller can also be more flexible. For example, such a controller could construct interfaces with the user's preferences in mind. If the user preferred to use their fingers to press buttons on the controller's screen, the user might ask that the buttons be made larger. If automatic machine translation was available, the user might be able to request a French or Japanese interface, with all the textual labels translated by the controller. The controller could also use the design of interfaces that are already in use as models for building interfaces to other unfamiliar appliances. When traveling, this might allow me to set the alarm on the unfamiliar clock in my hotel room. Instead of fumbling with the buttons and dials, I could simply pull out my PUC, download the specification from the alarm clock, and set the alarm using an interface similar to one I use for my alarm clock at home.

A separate self-programmed controller also allows appliances to be more expandable, even after being deployed. For example, if the phone system starts offering a new service, this might automatically appear as a new labeled button on the PUC, rather than requiring the user to remember to enter an arbitrary code like *69.

Another possible advantage is that the controller can serve as an "authentication token" and dynamically provide different interfaces to different classes of users. For example, some appliances might only provide certain controls to certain classes of users. 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 or volume. Thermostats in offices might allow anyone to change the temperature within certain ranges, but only certain people might be authorized to switch from heating to cooling.

APPROACH

Work has been done previously on the automatic generation of user interfaces, but most of these systems created only outlines of a interfaces and required significant work by the designer to make them usable [5][26]. This will not be suitable for our purposes, because our system is targeted at consumers instead of interface designers, and users cannot be expected to spend time to improve the generated interfaces. We feel that our system may be more successful than others for two reasons. Most importantly, the PUC only generates remote control interfaces, which are simpler than the interfaces used in generic PC applications. Second, we are basing the development of our generation technology on information collected about the features of high-quality hand-designed interfaces.

The remote control interfaces we are considering for this project are not as complicated as most PC applications because they require only a few types of widgets and do not use interaction techniques such as direct manipulation, which are more difficult to describe to an interface generator. Buttons, checkboxes, selection boxes, simple text-entry widgets, and static bitmaps are the only elements that have been needed to build our hand-designed interfaces, discussed in detail in later sections.

Our definition of a remote control interface may limit some of the appliances that a PUC could theoretically control. For example, one could imagine a picture-frame appliance with a digital display that was capable of showing an arbitrary picture. A PUC could allow a user to change the picture shown in the frame from a list of options, because this type of interaction is possible with buttons and selection boxes. The user could not draw an arbitrary picture on their controller and have it displayed in the frame however. We are confident that these restrictions will not prevent the PUC from being useful, because almost all appliances available today use buttons, dials, and simple displays as their only interface elements.

As a first step toward automatically generating remote control interfaces, we have hand-designed control panels for two appliances, evaluated them for quality, and attempted to extract the features of these control panels that contribute most to their usability.

We chose two common appliances as the focus of our hand-designed interfaces: the Aiwa CX-NMT70 shelf stereo with its remote control (see Figure 1) and the AT&T 1825 telephone/digital answering machine (see Figure 2). We chose these two appliances because both are common, readily available, and combine several functions into a single unit. The first author owns the Aiwa shelf stereo that we used, and the AT&T telephone is the standard unit installed in many offices at Carnegie Mellon. Aiwa-brand stereos seem to be particularly common, at least among our subject population, because ten of our twenty-five subjects owned Aiwa systems.

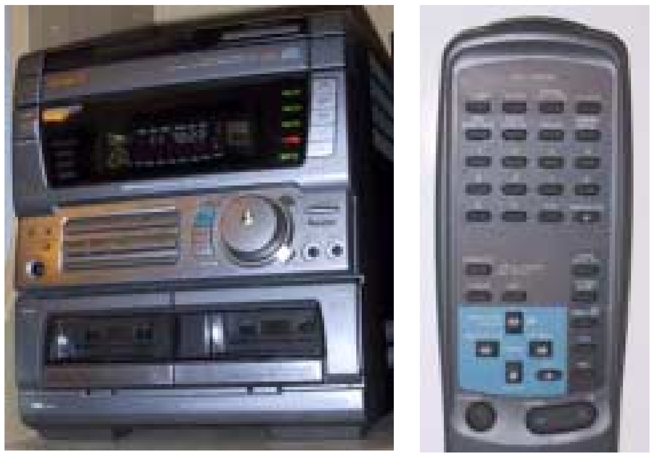
(a) (b)

Figure 1. a) The Aiwa CX-NMT70 shelf stereo used in our research. b) The remote control for the Aiwa stereo.

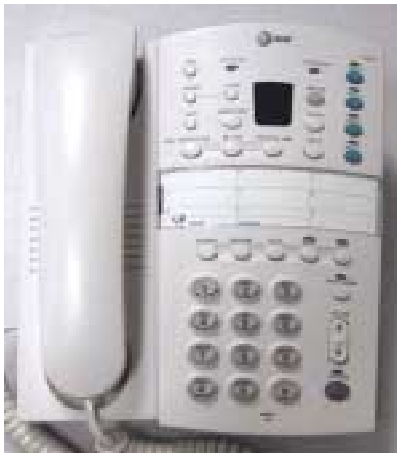

Figure 2. The AT&T 1825 office telephone/digital answering machine that we used in our research.

We created our hand-designed interfaces in two phases, initially on paper and later as full Visual Basic implementations on a Microsoft PocketPC. At each phase, we iteratively improved the interfaces with heuristic analyses [18] and performed a user study. The user study in each phase was dual-purpose: to compare our hand-designed interfaces with the interfaces on the actual appliances and to see what problems users had with the hand-designed interfaces. The next section describes these studies and their results.

USER STUDIES

Both studies that we conducted were between-subjects comparisons of the hand-designed interfaces that we created and the interfaces on the actual appliances. We measured the performance of the subjects using several metrics, including the time to complete a task, the number of errors made while attempting to complete a task, and how often external help was required to complete a task. The purpose of these studies was to discover how users performed using our hand-designed interfaces versus the interfaces of the actual appliances and discover what aspects of the hand-designed interfaces were difficult to use.

Procedure

Both studies were conducted between-subjects. When each subject arrived, they were asked to fill out a consent form and a two-page questionnaire about their computer background and remote control use. Then each subject worked through two task lists, with one of four possible combinations of the interfaces: actual stereo followed by handheld phone, actual phone followed by handheld stereo, handheld stereo followed by actual phone, and handheld phone followed by actual stereo. Thus, each subject saw one actual interface and one handheld interface, and one stereo interface and one phone interface, neither necessarily in that order. When finished, a final questionnaire was given that asked whether the actual appliance or handheld interface was preferred and for any general comments about the study and the interfaces.

Evaluation

In order to compare the interfaces for both appliances, task lists were created for the stereo and phone. Each list was designed to take about twenty minutes to complete on the actual appliance, and the same tasks were used for both the handheld and actual interfaces. About two-thirds of the tasks on both lists were chosen to be easy, usually requiring one or two button presses on the actual appliance. Some examples of easy tasks are playing a tape on the stereo, or listening to a particular message on the phone. The remaining tasks required five or more button presses, but were chosen to be tasks that a user was likely to perform in real life. These included programming a list of tracks for the CD player on the stereo, or setting the clock on the phone.

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 prototype interfaces were allowed to press the "Help" button, available in some form on every screen. This presented a scrollable screen of text, indexed by topic, that told the subject about the features of that particular interface.

The performance of each subject on both lists of tasks was recorded using three metrics: time to complete the tasks, number of missteps made while completing the tasks, and the number of times external help was needed to complete a task. The time to complete the tasks was measured from the press of the first button to the press of the button that completed the last task. External help is any use of the manual for an actual appliance or the help screen on the handheld.

For the purposes of this study, a misstep is defined as the pressing of a button that does not advance progress on the current task. Repeated pressings of the same button were not counted as additional missteps. Sometimes a subject would try something that did not work, try something else, and then repeat the first thing again. If the interface had given no feedback, either visibly or audibly, the repeated incorrect steps are not counted as additional missteps. No

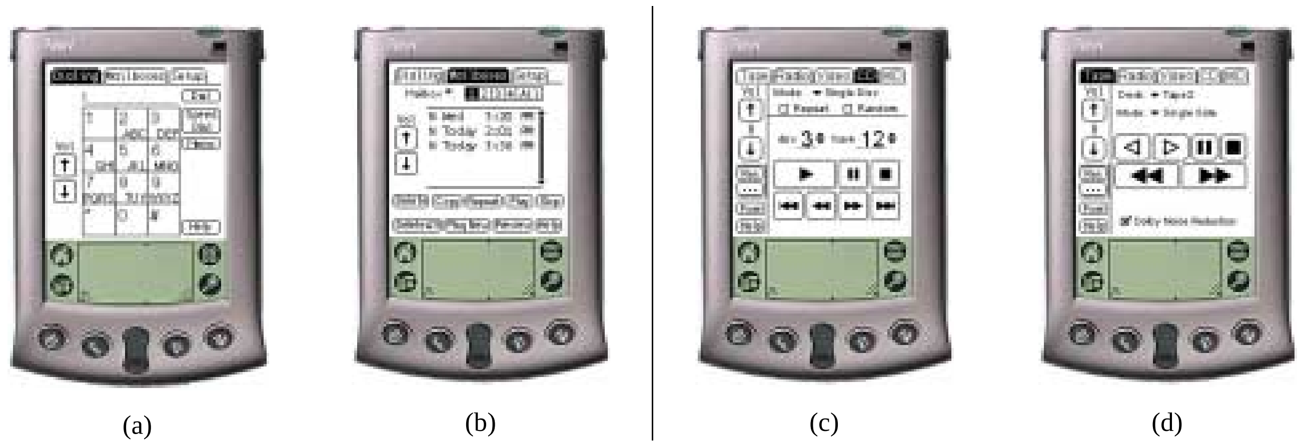

Figure 3. Paper prototypes of the phone (a-b) and stereo (c-d) interfaces for the Palm.

missteps are counted for a task after the user has requested external help.

STUDY #1

The first study compared the actual appliance interfaces to paper prototypes [25] of our hand-designed interfaces. Several examples of these interfaces are shown in Figure 3. The interfaces were also built to be functionally equivalent with the appliance they were meant to control, and are equally complex.

For the handheld portion of our experimental procedure, subjects were given a piece of paper that showed a picture of a Palm V handheld device with the remote control interface shown on the screen. Subjects were instructed to imagine that the picture was an actual handheld, and to interact with it accordingly. Whenever the subject tapped on an interface element on the screen, a new picture was placed over the old one to show the result of the action. If auditory feedback was required, such as when the subject pressed play on the CD panel of the stereo (Figure 3-c), the test administrator would verbally tell the subject what happened.

Participants

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 and seven owned Palm devices at the time of the study. Only 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.

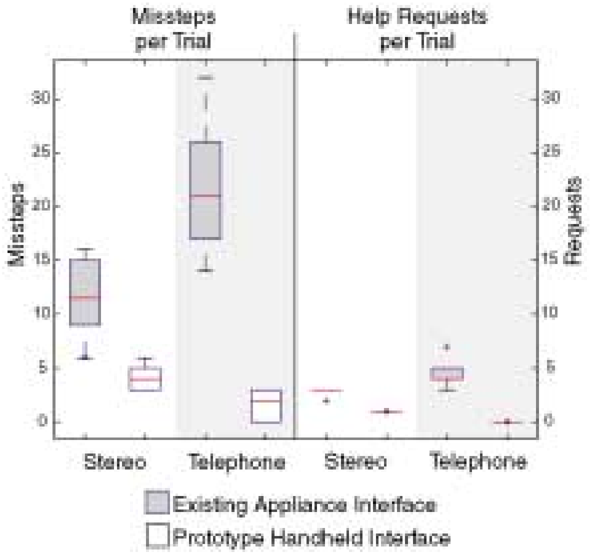

Figure 4. Box-plots showing the range of missteps and help requests (uses of external help) for each appliance and interface type.

Results

The results of the study indicate (all $p < 0.001$) that subjects made fewer missteps and asked for help less using the prototype handheld interfaces than using the actual appliances (see Figure 4). This indicates that the prototype handheld interfaces were more intuitive to use than the actual interfaces.

Time was not recorded for this study because we believed that delays created by the paper prototypes would dominate the time required to complete all the tasks. Even so, informal measurements suggested that subjects needed about one-half the time to complete all of the tasks using the prototypes as compared to the actual appliances.

Discussion

We found that users had great difficulty using the actual appliances, but were able to understand and operate the paper prototype interfaces with reasonable ease. One exception to this was found in the prototype stereo interface, which made use of the Palm's built-in menu system. None of our subjects navigated to screens that were only accessible through the menus without help, because they did not think to press the button that makes the menus visible. This was in spite of the fact that more than half used Palm devices regularly and were aware of the menu system.

Although the study was successful, we were concerned that the prototype interfaces benefited from the close interaction of the subject and the experimenter. In the paper prototype portion of the study, the experimenter provided all feedback to the user, including verbal hints when the user requested them. Because of these issues, we decided to conduct a new study with full implementations of the interfaces such that the experimenter would be a passive observer instead of an active participant.

STUDY #2

The second study improved upon on the first study by replacing the paper prototypes with working prototypes. We created interfaces for the stereo and phone that run on a Microsoft PocketPC using Visual Basic. These interfaces were based on the prototype interfaces for the Palm, but were modified to conform to the interface conventions of the PocketPC (see Figure 5). We chose the PocketPC instead of the Palm because of the availability of Microsoft's eMbedded Visual Basic tool, which made the implementation relatively painless.

Unfortunately, it was not possible to use the PocketPC to actually control either of our appliances, but we still wanted the subjects to receive feedback to their actions in a manner that was consistent with the appliances, without involving the experimenter. We chose to simulate control using wireless communication from our handheld to a laptop. The laptop was connected to external speakers, and it generated auditory feedback that was consistent with what would be expected if the PocketPC were actually controlling either of the appliances.

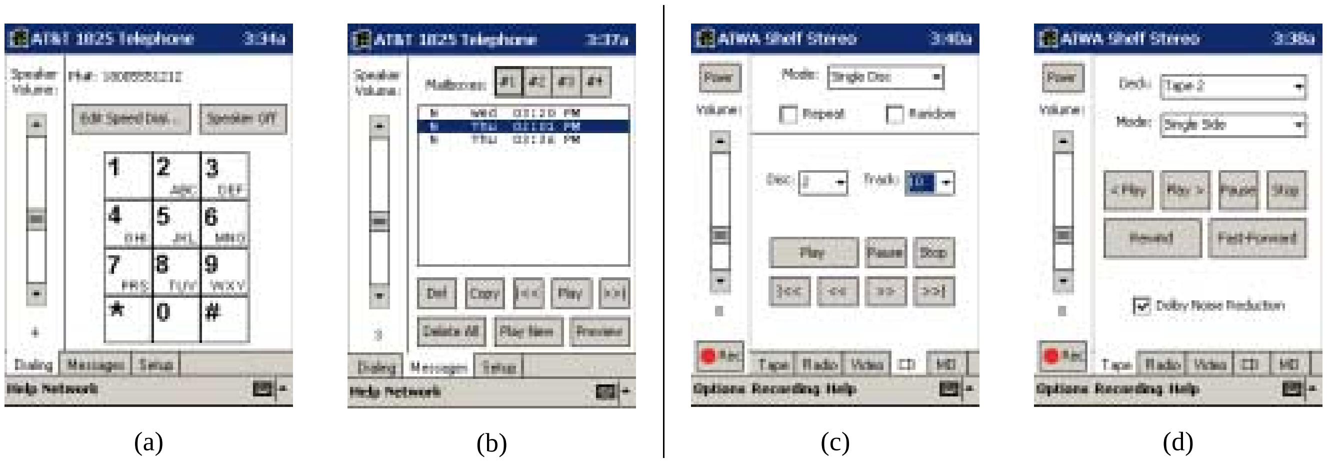

Figure 5. Screenshots of the implemented phone (a-b) and stereo (c-d) interfaces for the PocketPC.

Because of the complexity of both appliances, the PocketPC interfaces required more than fifty hours of design and implementation effort to create. The PocketPC implementations were improved over several iterations using a combination of heuristic analysis and think-aloud studies with pilot users. The results of these analyses are included in the discussion of this study, below.

One very important issue with the PocketPC interfaces was their use of several conventions that are specific to the PocketPC operating system. In particular, there is a standard OK button for exiting dialog boxes that is displayed in the top right corner of the screen. Users in pilot tests did not discover this feature, and thus were unable to exit from certain screens in the interface. Because one goal of the personal universal controller is to use the conventions of the controlling device to guide interface generation, we chose not to change the interfaces. Instead, we created a tutorial program that we presented to subjects before they begin using the PocketPC interface. The tutorial covers the OK button, text-entry, and the location of the menu bar, which is at the bottom of the screen instead of the top.

Participants

Twelve students from Carnegie Mellon volunteered to participate in the study, in response to an advertisement posted on a high-traffic campus newsgroup. The advertisement specifically requested people with little or no knowledge of handheld computers. Subjects were paid US$7 for their participation in the study, which took between thirty and forty-five minutes to complete. Eight men and four women participated, with a median age of 22 and an average of five years experience using computers. Subjects self-rated their skill at using computers for everyday tasks and their knowledge of handheld computers on a seven-point Likart scale. On average, subjects rated their knowledge of handhelds three points less than their skill with everyday computers (an average of 5.5 for everyday skill and 2.5 for handheld knowledge). Half the group owned Aiwa-brand stereos and two had AT&T digital answering machines.

Results

The results of the study indicate that subjects performed significantly better ($p < 0.05$ for all) using the handheld interfaces in all three metrics. Table 1 shows the data collected for each of the three metrics on the stereo and phone interfaces. Figure 6 and Figure 7 show box-plots comparing the handheld and actual interfaces for each metric on the stereo and phone respectively.

For both appliances, users of the actual interfaces took about twice as long, needed external help five times more often, and made at least twice as many mistakes as users of the handheld interfaces.

Discussion

The results of the second study are very similar to those of the first. Most of our subjects did not need to use external help to complete tasks using the handheld, and those that did use help only used it once. This compares to each subject's average of 3.6 uses of help for the actual stereo and 4.3 uses for the actual phone. Poor labeling, insufficient feedback, and the overloading of some buttons with multiple functions can account for this large difference on the actual appliances.

The worst examples of poorly labeled buttons and overloaded functions were found on the AT&T phone. This phone has several buttons that can be tapped quickly to activate one function and be pressed and held to activate another function. There is no text on the telephone 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.

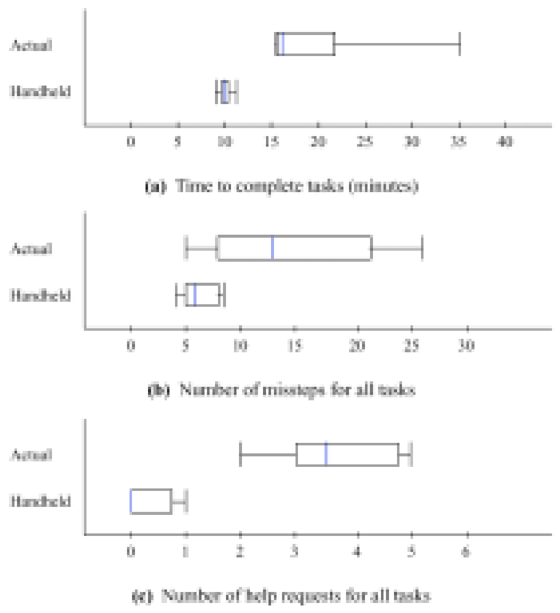

Figure 6. Box-plot comparisons of the stereo interfaces.

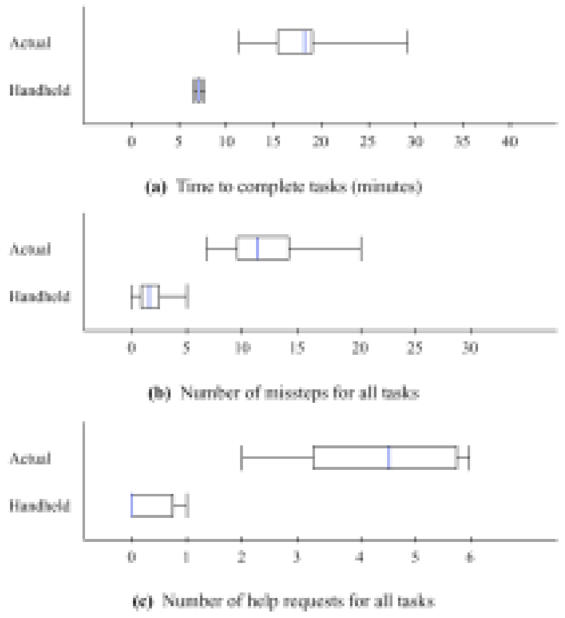

Figure 7. Box-plot comparisons of the phone interfaces.

Table 1. The raw data collected in the second study. Subject numbers were assigned randomly and do not reflect the order in which a subject participated in the study.

	Stereo						**Telephone**					
	Actual			*Handheld*			*Actual*			*Handheld*		
Subj.	**Time**	**Missteps**	**Help**	**Time**	**Missteps**	**Help**	**Time**	**Missteps**	**Help**	**Time**	**Missteps**	**Help**
142	34:43	26	5	-	-	-	-	-	-	07:50	1	1
246	15:36	7	3	-	-	-	-	-	-	07:01	0	0
269	22:50	10	5	-	-	-	-	-	-	06:52	5	1
323	16:11	23	4	-	-	-	-	-	-	06:49	3	0
391	15:22	5	3	-	-	-	-	-	-	07:03	3	0
591	15:53	16	2	-	-	-	-	-	-	06:14	1	0
57	-	-	-	09:09	4	0	18:54	12	6	-	-	-
106	-	-	-	09:27	8	1	11:04	7	2	-	-	-
296	-	-	-	10:24	8	0	14:42	15	3	-	-	-
412	-	-	-	10:37	5	1	19:03	9	5	-	-	-
438	-	-	-	09:47	7	0	17:25	20	4	-	-	-
682	-	-	-	11:31	5	0	29:11	11	6	-	-	-

The phone also suffered from an underlying technical separation between the telephone and the answering machine functions. None of the buttons on the phone can be used with the answering machine. Even the numeric codes must be set using arrow buttons rather than the phone keypad. All but one subject tried to use the keypad buttons to set the code. The exception had used a similar AT&T phone in the past.

All of these problems could be avoided in the handheld interfaces, because there was room for labels that were more descriptive and certain multi-step functions could be put on a separate screen or in a wizard. Using different screens to separate infrequently used or complex functions can also be problematic, however. Other buttons or menu items must be provided so that the user can navigate between screens, and the labels for these navigation elements must describe the general contents of the screen that they lead to. This was particularly a problem for the handheld stereo interface, which has more than ten screens and is more complex than the handheld phone interface. Many of the screens are accessible through the menu bar at the bottom of the screen. Subjects in the study and think-aloud participants before the study were very tentative about navigating the menus to find a particular function. In tasks that required the subject to navigate to a screen from the menu bar, the subject commonly opened the correct menu, closed the menu, did something wrong on the current screen, and then opened the menu again before finally picking the correct item.

The handheld stereo interface had other problems as well. In particular, we found that the record function was difficult to represent in the interface because it was associated with tapes but needed to be available in all of the stereo's five playback modes: tape, radio, CD, etc. Although a record button was available on every screen (see Figure 5-c), many subjects would get confused and incorrectly switch to the tape mode instead of pressing the record button. The red circle next to the text label on the "Rec" button was added to make the button more visible, because we thought that people tried the tape mode because they did not see the record button. This change seemed to have little effect.

This type of dual-associated function presents two difficulties for future personal universal controllers. It is difficult to communicate to users how they should use the function and it complicates the specification language that the PUC will build interfaces from. Without elements like the record button, it would be possible to represent the functions of an appliance with a hierarchical tree. For a stereo, the root node would be the power state with two children that represent when the unit is on or off. The "power on" child might have five children representing each of the different playback modes, and further branching would specify the states of each mode. The record button does not fit at any of these levels, because it affects the stereo globally but with different local effects. For example, recording from radio is different from recording from CD, but in both cases the same tape is being recorded onto. Using a graph to represent the relationships between appliance functions can solve this problem, but may make it more difficult to infer functional groups.

The prototype interfaces showed that finding functional groups is key to constructing a good interface. These groups define how elements are placed relative to each other, and which elements can be separated across multiple screens. The different screens of the tab components are the best examples of grouping in our prototype interfaces (see Figure 4 and Figure 5). Grouping is also used to separate the mode, random, and repeat elements from the rest of the elements of the stereo CD player interface (see Figure 5-c). These elements are used in all of the CD player's modes, while the other components are only used in half the modes. We are currently exploring ways to make these grouping easier to infer from the specification language.

Unfortunately, the groups cannot be specified directly because their members may vary between target platforms. For example, on a device with a small screen it might be necessary to separate the display of the current disc and track from the controls for playing a CD. It would not be appropriate if the PUC separated the play and stop buttons however. A challenge will be finding a way to embed this knowledge in the specification language without defining every possible group.

Our prototype interfaces also uncovered some issues that will make the implementation of a type system within our specification language difficult. We had assumed that each element in the specification language would have a type, and that the type of an element would indicate which widget should represent it. This is problematic because sometimes the widget suggested by the type may conflict with the widget that would chosen to match the conventions of the target platform. For example, the standard volume widget on the Palm is a selection box with four choices: Off, Low, Medium, and High. This would not be an appropriate choice for the volume widget on the stereo, because the type of the volume control element for the stereo is an integer between 0 and 30. Indeed, we chose not to use the Palm's volume widget in our prototypes (see Figure 3-c), but generalizing that choice could prove more difficult.

Sometimes it may also make sense to have multiple elements of the specification language be instantiated as a single widget. The prototype PocketPC interface for the phone does this, by using the same scrollbar to set the volume of the speakerphone and the handset, depending on which is currently in use (see Figure 5-a). The interface generator may be able to infer this if it knew that the elements were never used together and their types were similar. Matching types exactly will not work though. In the phone example, the handset volume ranges from 1-4 but the speakerphone volume ranges from 1-8.

The opposite of this situation may also arise, where a single element in the specification language may be instantiated several times in the interface. The volume control in the prototype stereo interface is shown on the main screen and also duplicated in a dialog box that we expect to be used frequently. This gave the user easy access to the volume when they needed it, even if they had obscured the normal volume control with the dialog box.

FUTURE WORK

Our next goal is to design the specification language that will be used to guide the automatic generation of the remote control interfaces. In addition to using the knowledge we gained from our prototype studies, we will also be collaborating with visual interface designers and a project team from Maya Design Corp., http://www.mayadesign.com. The team from Maya is specifically interested in the architecture that underlies the automatic generation of interfaces. In addition to the specification language, this also includes the programming interface that will separate the automatic generation software from the specifics of each target platform.

RELATED 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 [9]. However, their research seems to have focused on the system and infrastructure issues rather than how to create the user interfaces. Hodes's later paper describes the "rvic" system [10] 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 [6] 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 [2], but apparently has not yet addressed this issue. Another IBM project [4] 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 [20] 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 [1], 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 [30] system, Rekimoto's many systems [22][23][24], and our Pebbles system [12][13][14][15]. 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 [28]. 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 [11][29] and creating complete interfaces in a limited range [19][7][28]. The ITS system from IBM was used to create all the screens for the information kiosks at the EXPO'92 worlds fair [31][32]. Of particular note is the layout algorithm in the DON system that achieved a pleasing, compact, and logical placement of the controls [11]. Other systems focused on the initial creation assuming a user would edit the resulting user interface [5][26]. We plan to extend these results to create panels of controls on handhelds of significantly different properties.

ACKNOWLEDGMENTS

This work was conducted as a part of the Pebbles [15] project, led by the second author. Marc Khadpe did a portion of the work on the prototype phone interface as a part of a summer independent study project. We would also like to thank Mike Schneider, Anupriya Ankolekar, and Ryan Baker for giving their time to participate as pilot subjects for the second study. This work was funded in part by the Pittsburgh Digital Greenhouse and in part by a National Science Foundation Graduate Fellowship.

REFERENCES

1. Abrams, M., *et al.* "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/
2. Banavar, G., *et al.* "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/
3. Brouwer-Janse, M.D., *et al.* "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.
4. Eustice, K.F., *et al.* "A Universal Information Appliance," *IBM Systems Journal.* 1999. **38**(4). pp. 575-601. http://www.research.ibm.com/journal/sj/384/eustice.html
5. Foley, J.D., *et al.* "A Knowledge-Based User Interface Management System," *Human Factors in Computing Systems,* Proceedings SIGCHI'88. Washington, D.C., May, 1988, 1988. pp. 67-72.
6. Fox, A., *et al.* "Integrating Information Appliances into an Interactive Workspace," *IEEE Computer Graphics and Applications*, May/June 2000,: pp. 54-65.
7. Frank, M.R. and Foley, J.D. "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.
8. Haartsen, J, *et al.* "Bluetooth: Vision, Goals, and Architecture," *ACM Mobile Computing and Communications Review.* 1998. **2**(4). pp. 38-45. Oct. www.bluetooth.com.
9. Hodes, T., *et al.* "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.
10. Hodes, T., *et al.* "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
11. Kim, W.C. and Foley, J.D. "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.
12. Myers, B.A., Stiel, H. and Gargiulo, R. "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
13. Myers, B.A., Lie, K.P., and Yang, B. "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.
14. Myers, B., *et al* "Extending the Windows Desktop Interface With Connected Handheld Computers," *4th USENIX Win-*

dows Systems Symposium. 2000. Seattle, WA: pp. 79-88. http://www.cs.cmu.edu/~pebbles/.

15. Myers, B.A., *et al.* “Using Hand-Held Devices and PCs Together,” *ACM Communications of the ACM.* 2001b. p. To appear.
16. Narayanaswami, C. and Raghunath, M.T. “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
17. Nichols, J. “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.
18. Nielsen, J. *Usability Engineering*. Boston, Academic Press. 1993.
19. Olsen Jr., D.R. “A Programming Language Basis for User Interface Management,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 171-176.
20. Olsen Jr., D.R., *et al.* “Cross-modal Interaction using Xweb,” in *Proceedings UIST’00: ACM SIGGRAPH Symposium on User Interface Software and Technology*. 2000. San Diego, CA: pp 191-200.
21. Philips. *Pronto Intelligent Remote Control*. Philips Consumer Electronics. 2001. http://www.pronto.philips.com/
22. Rekimoto, J. “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.
23. Rekimoto, J. “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.
24. Rekimoto, J and Saitoh, M. “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.
25. Rettig, M., “Prototyping for Tiny Fingers,” *Communications of the ACM*, April 1994,: pp. 21-27.
26. Singh, G. and Green, M. “A High-Level User Interface Management System,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 133-138.
27. Sun. *Jini Connection Technology*. Sun Microsystems. http://www.sun.com/jini/. 2000.
28. Szekely, P., Luo, P. and Neches, R. “Beyond Interface Builders: Model-Based Interface Tools,” *Human Factors in Computing Systems,* Proceedings INTERCHI'93. Amsterdam, The Netherlands, Apr, 1993. pp. 383-390
29. Vander Zanden, B. and Myers, B.A. “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.
30. Want, R., *et al.* “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.
31. Wiecha, C., Bennet, W., Boies, S. and Gould, J. “Generating user interfaces to highly interactive applications,” *Human Factors in Computing Systems,* Proceedings SIGCHI'89. Austin, TX, Apr, 1989, 1989. pp. 277-282.
32. Wiecha, C., *et al*. “ITS: A Tool for Rapidly Developing Interactive Applications,” *ACM Transactions on Information Systems.* 1990. **8**(3). pp. 204-236