2003

Studying the Use of Handhelds to Control Smart Appliances

Jeffrey Nichols
Carnegie Mellon University

Brad A. Myers
Carnegie Mellon University

Studying The Use of Handhelds to Control Smart 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— **Today's complex appliances are plagued by difficult-to-use interfaces. In many cases, consumers use only a few of the many features on their appliances because the more complex features are hidden by confusing interfaces. This problem can only get worse as appliances get smarter, become more complex, and are subject to more demands by their users. This paper presents two studies that compare the accuracy and speed of real users controlling two common appliances, a stereo and a telephone/answering machine, using two different interaction techniques. Our studies found that people using an appliance interface presented on a handheld computer performed the same set of tasks in half the time while making half the errors as compared to using the appliance's built-in control panel. These studies are motivating us to build a generic remote appliance control system that we call the *personal universal controller*.**

1 Introduction

Appliances available today are becoming increasingly complex. Unfortunately for users, the trend has been that as these appliances get more complex, their interfaces become harder to use [1]. This is occurring in part because it is cheap for manufacturers to embed powerful processors and add new features, but it is expensive to design a high quality user interface that accommodates all of the new features.

Making user interfaces for the smart appliances of the future will be even more difficult and expensive for several reasons: 1) Smart appliances will be even more complex than today's appliances, and thus the interface design task will be more difficult. 2) Smart appliances may be able to update their functionality, perhaps by downloading code for new features over the Internet. The interface for the smart appliance will need to be flexible enough to present these new features in a usable way. 3) Users will want to remotely control their appliances from great distances, such as controlling a device at home from the workplace. 4) Users may want to combine the interfaces of multiple smart appliances into a single task-specific user interface. For example, the user might want to have a set of controls for their living room that combine the interfaces of their DVD player, television, and stereo system.

We are developing the *personal universal controller* (PUC), a technology that will solve these problems by letting users interact with intermediary interfaces as though they were using remote controls. We imagine that PUCs will run in a variety of locations through multiple interface modalities, perhaps as graphical interfaces on personal digital assistants (PDAs) or speech interfaces on hidden household computers. Unlike the remote controls of today, the PUC is *self-programming*. This means that the PUC engages in a two-way exchange with the appliance, downloading a specification of the appliance's functions and then automatically generating a high-quality user interface. The two-way communication between the PUC and the appliance continues as the user interacts with the interface, allowing the PUC to send command signals to the appliance and show feedback of the appliance's state.

The PUC will solve several of the problems that we anticipate for the interfaces of smart appliances. Since appliance interfaces are automatically generated, the appliance may upgrade its features at any time and the interface will update to reflect the new features. A PUC interface can also be used anywhere, provided a network connection can be made to the appliance. This allows a PUC to remote-control appliances from a distant location. Remote control is made easier because feedback on the appliances' states is provided by the two-way communication, which means that users will not be blindly controlling appliances.

We also believe that the PUC will be able to deal with the complexity of an appliance's features better than an interface on the appliance itself, because PUC interfaces are being generated in a richer interface environment than is possible to provide on an appliance. In this paper we present two studies that compare user accuracy and speed using interfaces on a PDA versus using the interfaces on the actual appliances. Our studies show that users were twice as fast and made half the errors when using the PDA interface as compared to the actual appliances. This difference can be attributed to the lack of flexibility given the designers of the actual appliances.

This paper begins by describing some work related to the PUC project. We then describe our approach to creating a system that automatically generates high quality interfaces. We continue by describing the user studies, their results, and discuss how these results are helping us build a system that automatically generates user interfaces. We conclude by briefly describing the current state of the PUC project and our directions for the future. Prior papers have described the PUC system in more detail [4, 5], but this is the first description of the user studies that motivated our work.

2 Related Work

A number of research groups are working on controlling appliances from handhelds. The Stanford ICrafter [7] is a framework for distributing appliance interfaces that supports automatic generation of interfaces. The focus of this project was on the infrastructure however, and [7] shows only one low quality automatically generated interface. The XWeb project [6] automatically generates interfaces with complex layouts, however their interfaces do not seem to handle modes well, which is an important part of building interfaces for appliances. The importance of handling modes will be seen later in this paper, in the discussion of the user studies.

Work has also been done previously on the automatic generation of generic user interfaces for any application, but most of these systems created only outlines of interfaces and required significant work by the designer to make them usable [2, 8]. 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.

The INCITS V2 [9] standardization effort is developing the Alternative Interface Access Protocol (AIAP) to help people with disabilities interact with everyday appliances. This work is very similar to our PUC protocol, and we are now collaborating closely with this group. Research results from our PUC project have already significantly affected the direction of the V2 effort.

3 Approach

Automatically generating high quality user interfaces is a hard problem, as shown above in the related work section. As a first step toward automatically generating remote control interfaces, we hand-designed control panels for two appliances, evaluated them for quality, conducted the two user studies reported here, and then attempted to extract the features of these control panels that contribute most to their usability [5]. This approach allows us to understand the features that a high quality remote control interface will have, and then apply them in our interface generator software [4].

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 1a) and the AT&T 1825 telephone/digital answering machine (see Figure 1b). 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.

4 User Studies

Both studies that we conducted were between-subjects comparisons of the hand-designed PDA 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.

4.1 Procedure

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 with two interfaces in one of four possible combinations to control for order effects. Each subject saw one actual interface and one handheld interface, and one stereo interface and one phone interface, neither necessarily in that order. For each interface, the user was asked to work through a set of tasks. When finished, a final questionnaire was given that asked whether the actual appliance or PDA interface was preferred and for any general comments about the study and the interfaces.

4.2 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 time 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

(a) (b)

Figure 1. a) The Aiwa CX-NMT70 shelf stereo and b) the AT&T 1825 office telephone/digital answering machine used in our research.

(a) (b)

Figure 2. Interfaces for the phone (a) and stereo (b) hand-designed for the Palm. These interfaces are paper prototypes.

"Help" button, available in some form on every screen. On the paper interfaces for the first study, a verbal hint was given, whereas the Visual Basic interfaces for the second study presented a scrollable screen of text, indexed by topic.

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 PDA, or any verbal hint from the experimenter.

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

5 Study #1

The first study compared the actual appliance interfaces to paper prototypes of our hand-designed interfaces. Two examples of these interfaces are shown in Figure 2. The interfaces were also designed to be functionally equivalent with the appliance they were meant to control.

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 displaying the remote control interface. 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 (see Figure 2b), the test administrator would verbally tell the subject what happened.

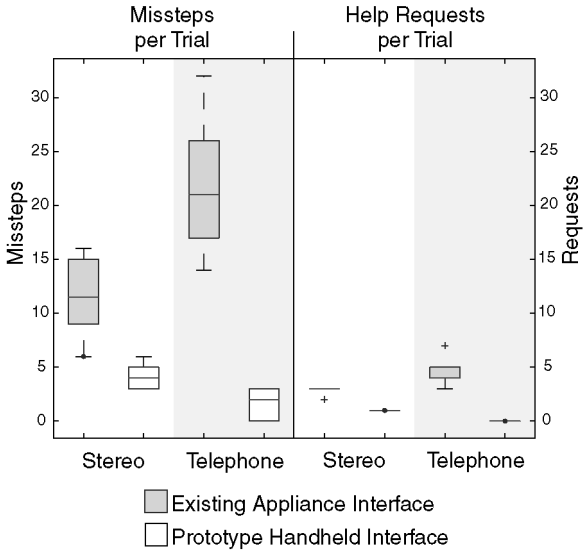


Figure 3. Box-plots showing the range of missteps and help requests (uses of external help) for each appliance and interface type.

5.1 Participants

Thirteen Carnegie Mellon graduate students volunteered to participate in this study, five female and eight male. All subjects were enrolled in the School of Computer Science, and all had significant computer experience. 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.

5.2 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 3). 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.

5.3 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 so that the experimenter would be a passive observer instead of an active participant.

6 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 4). 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.

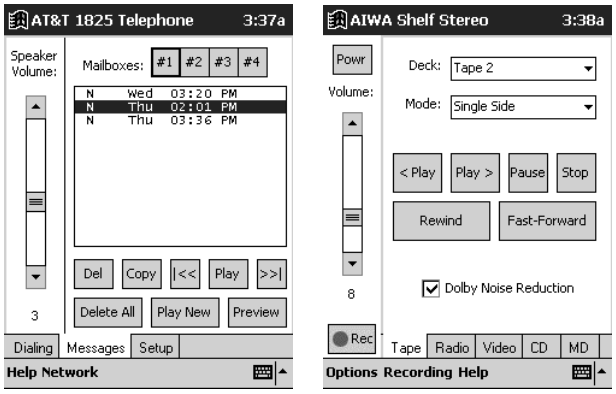
Unfortunately, it was not possible to use the PocketPC to actually control either of our appliances, but we still wanted the subjects to receive feedback from their actions in a manner that was consistent with the appliances, without involving the experimenter. We chose to simulate control using wireless communication from our PDA 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.

Because of the complexity of both appliances, the PocketPC interfaces required more than fifty hours of design and implementation effort to create. They were improved over several iterations using a combination of heuristic analysis and think-aloud studies with pilot users.

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 of our goals 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 began the study. 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 as on desktop computers.

6.1 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.



(a) (b)

Figure 4. Interfaces for the phone (a) and stereo (b) hand-designed for the PocketPC. These interfaces were actually implemented in Microsoft's eMbedded Visual Basic.

6.2 Results

The results of the study indicate that subjects performed significantly better ($p < 0.05$ for all) using the PDA interfaces in all three metrics: time to complete the tasks, number of help requests, and number of missteps. Note that we were able to measure time in this study because there was no longer time overhead from shuffling papers. Figure 5 shows 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 PDA interfaces.

6.3 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.

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 were avoided in the PDA 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. 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 PDA 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 4b), 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 after pilot testing 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, however.

The prototype interfaces showed that finding functional groups is important for 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 2 and Figure 4). 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 2b). These elements are used in all of the CD player's modes, while the other components are only used in half the modes.

Unfortunately, the visual groups cannot be specified explicitly 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. We have found that we can specify this grouping information to the interface generators by combining a tree structure of the appliance's functions with dependency information. Dependency information defines when each function will be active in terms of the other functions. Knowledge of dependencies is particularly useful for determining visual groups, as it can be used to separate

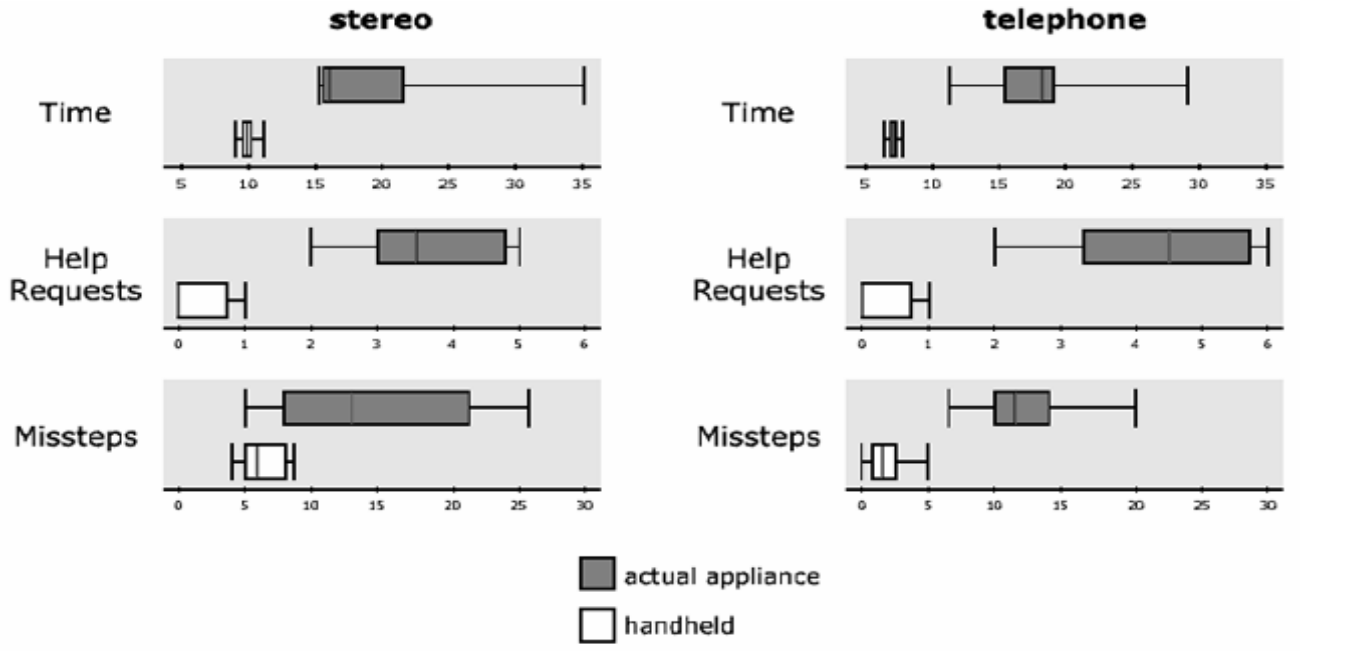


Figure 5. Box-plots of results from the second user study.

controls that are only available in certain modes, as we saw in the stereo CD player interface.

The prototype interfaces also showed that feedback is important. One important way the interfaces provided feedback was by disabling, or graying out, a control so the user could tell when that function was not available. Many of the errors users made with the actual appliance interfaces occurred when they pressed buttons for functions that were not currently available. This is another area where dependency information is helpful, because it defines exactly when the control(s) for a function should be disabled.

One problem for interface generators is dealing with the conventions that we are familiar with in our daily lives. For example, both the actual phone and our PDA phone interface used the standard layout for a phone dialing pad (see Figure 2a). These conventions should not be specified as a part of the functional description of the appliance however, because they are not always applicable. The dialing pad convention does not make any sense for a speech interface, for example. We plan to solve this problem by defining high-level types in our language that may not be understood by every interface generator. Those generators that do understand the type can apply the appropriate convention, and those that do not will be able to query the appliance further to understand the primitive components of the high-level type.

Another problem for interface generators is determining when it makes 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 4a). One solution would be to use this technique when two functions have similar types and are never used together. We plan to investigate the generalness of this solution as a part of our future work.

7 Current Status and Future Work

From the lessons of these studies, we developed a set of requirements for any system that automatically generates remote control interfaces [5]. We then designed a language for specifying the functions of an appliance and built two interface generators that automatically create graphical and speech user interfaces [4].

Our system can also control real appliances using two-way communication, such as a Sony digital camcorder, an Audiophase shelf stereo, a Mitsubishi VCR, X10 lighting, and the software media players Winamp and Windows Media Player. We communicate with these appliances by building appliance adaptors, which are custom hardware and software that translate the proprietary protocol of the appliance to our PUC protocol.

Much work remains to be done on this project. We would like our interface generators to create more aesthetic interfaces, and to take into account the preferences of the user during generation. We would also like our generators to ensure consistency across interfaces for similar appliances. In this way, for example, users could leverage their knowledge of a VCR they have used in the past to use a different VCR that they have never seen. We are also planning to integrate our system with some of the appliance control protocols that are emerging, such as UPnP and HAVi.

8 Conclusion

This paper has described two user studies that showed users were able to control an appliance using PDA interfaces twice as fast, while making half the errors as compared to using actual appliance interfaces. These studies show that PDAs, with their rich interface capabilities, are a promising direction to explore for appliance control. The prototype interfaces created for these studies provide insights into the important aspects of making highly usable control panels. The results of these studies suggest that it may be possible to make complex appliances usable with the PUC approach.

Acknowledgments

This work was conducted as a part of the Pebbles project [3]. Marc Khadpe did a portion of the work on the prototype phone interface as a part of a summer independent study project. This work was funded in part by grants from NSF, Microsoft and the Pittsburgh Digital Greenhouse, and equipment grants from Symbol Technologies, Hewlett-Packard, and Lucent. The National Science Foundation funded this work through a Graduate Research Fellowship for the first author, and under Grant No. IIS-0117658. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] Brouwer-Janse, M.D., *et al.* "Interfaces for consumer products: "how to camouflage the computer?"" in *CHI'1992: Human factors in computing systems.* 1992. Monterey, CA: pp. 287-290.

[2] Kim, W.C. and Foley, J.D. "Providing High-level Control and Expert Assistance in the User Interface Presentation Design," in *Human Factors in Computing Systems.* 1993. Amsterdam, The Netherlands: pp. 430-437.

[3] Myers, B.A., "Using Hand-Held Devices and PCs Together." *Communications of the ACM*, 2001. **44**(11): pp. 34-41. http://www.cs.cmu.edu/~pebbles/papers/pebblescacm.pdf.

[4] Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M. "Generating Remote Control Interfaces for Complex Appliances," in *UIST 2002.* 2002. Paris, France: pp. 161-170. http://www.cs.cmu.edu/~pebbles/papers/PebblesPUCuist.pdf.

[5] Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Shriver, S. "Requirements for Automatically Generating Multi-Modal Interfaces for Complex Appliances," in *ICMI.* 2002. Pittsburgh, PA:

[6] 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.

[7] Ponnekanti, S.R., *et al.* "ICrafter: A service framework for ubiquitous computing environments," in *UBICOMP 2001.* 2001. Atlanta, Georgia: pp. 56-75.

[8] Wiecha, C., *et al.*, "ITS: A Tool for Rapidly Developing Interactive Applications." *ACM Transactions on Information Systems*, 1990. **8**(3): pp. 204-236.

[9] Zimmermann, G., Vanderheiden, G., Gilman, A. "Prototype Implementations for a Universal Remote Console Specification," in *CHI'2002.* 2002. Minneapolis, MN: pp. 510-511.