Demonstrating the Viability of Automatically Generated User Interfaces

Jeffrey Nichols*
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
jwnichols@us.ibm.com

Duen Horng Chau, Brad A. Myers
Human Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{dchau, bam}@cs.cmu.edu

ABSTRACT
We conducted a user study that demonstrates that automatically generated interfaces can support better usability through increased flexibility in two dimensions. First, we show that automatic generation can improve usability by moving interfaces that are constrained by cost and poor interaction primitives to another device with better interactive capabilities: subjects were twice as fast and four times as successful at completing tasks with automatically generated interfaces on a PocketPC device as with the actual appliance interfaces. Second, we show that an automatic generator can improve usability by automatically ensuring that new interfaces are generated to be consistent with users' previous experience: subjects were also twice as fast using interfaces consistent with their experiences as compared to normally generated interfaces. These two results demonstrate that automatic interface generation is now viable and especially desirable where users will benefit from individualized interfaces or where human designers are constrained by cost and other factors.

Author Keywords
Automatic interface generation, handheld computers, personal digital assistants, mobile phone, personal universal controller (PUC), consistency, Pebbles

ACM Classification Keywords
D.2.2 Design Tools and Techniques: User interfaces – automatic generation. H.5.2. User Interfaces: Graphical user interfaces (GUIs).

INTRODUCTION
Researchers have been producing systems for automatically generating user interfaces for more than two decades. Two initial motivations for previous work were to better separate the user interface component from the input/output layer and to help programmers without any design training produce high-quality user interfaces. With the development of better interface abstractions and the increased availability of trained interface designers, these techniques for automatically generating interfaces were generally not adopted [7].

Recently, however, research into automatic generation has experienced a renaissance with several new systems offering improved generation algorithms and new user customization features. This work is motivated in several ways:

- The increasing diversity of computing devices providing a user interface, from handheld computers and tablet PCs to mobile phones and wristwatches, requires multiple user interfaces to be constructed for each application. Automatic generation can allow applications to be quickly ported to different platforms [2, 6, 8].
- For certain devices, especially office appliances and consumer electronics, it is economical for manufacturers to include many complex functions but expensive to provide a high-quality user interface [1]. One solution is to automatically generate the appliance interface on another device, such as a handheld computer or mobile phone, which can provide a higher quality user interface for all of the appliance's complex functions [8].
- There are many users with different backgrounds, goals, and capabilities using today's user interfaces, and each may benefit if his or her interfaces are specifically designed to take individual needs into account [2-4, 10]. It is impractical for human designers to create a different interface for each individual user, but an automatic interface generator can easily do this. For example, users with tremor could benefit from interfaces designed to support their particular type of tremor [4].

In order for new work in automatic generation to be adopted, it will need to improve on interfaces that are cur-

CHI 2007, April 28–May 3, 2007, San Jose, California, USA.

* This work was conducted while the first author was affiliated with the Human Computer Interaction Institute at Carnegie Mellon University.

a. HP printer without consistency

b. Canon printer without consistency

c. HP printer consistent with Canon printer

d. Canon printer consistent with HP printer

Figure 1. PocketPC interfaces generated by the Personal Universal Controller (PUC) for the two all-in-one printers discussed in this paper.

rently available today without increasing the development and manufacturing costs of products that might use these technologies. In particular, the cost of writing the abstract specification used to automatically generate an interface must not exceed the cost of manually designing and implementing an interface using today's technology.

In this paper, we present a user study that examines the usability of interfaces automatically generated by the Personal Universal Controller (PUC) system [8] (see Figure 1). We show that the PUC is able to produce interfaces superior to those available today at approximately the same cost by moving the interface from the appliance to a handheld device that users are already carrying. The results show that users of the PUC interfaces presented on a PocketPC device were *twice as fast* and *four times more successful* than users of the existing interfaces for a set of eight independent tasks of varying difficulty.

We also investigate the capability of automatically generated interfaces to provide benefits beyond what is practical for human designers to do. Specifically, we show that an automatically generated interface can be easier to learn if it is generated to be *consistent* with users' previous experiences. This functionality is supported by the PUC's Uniform layer [10]. To analyze this, after the subjects tried to perform the eight tasks using one interface, we trained them on the best way to do those tasks using that interface. Then we asked them to perform the same tasks on a different interface that provided similar functionality. We found that users were *twice as fast* when the second interface is generated by Uniform to be consistent with the first interface, as compared to when the second interface is generated with the normal PUC system with Uniform's features disabled.

Our study compares interfaces for two different all-in-one printer appliances. We focus on appliance interfaces because the PUC system is designed specifically for moving the interfaces from computerized appliances to a handheld device, such as a PDA or mobile phone. The two all-in-one printers we used are a Hewlett-Packard (HP) Photosmart 2610 with a high-quality interface including a color LCD, and a Canon PIXMA MP780 with a few more features and an interface that turned out to be harder to learn than the HP. These two represented the top-of-the-line consumer models from these manufacturers and the most complex all-in-one printers available for home use at the time of their purchase. We chose the all-in-one printers as our appliances in the study for several reasons:

- Complex appliances are typically more difficult to use than trivial ones and we wanted to test the PUC with appliances that would be challenging for its generation algorithms. We found that all-in-one printers were at least as complicated, if not more so, than many of the other appliance types that we have explored (containing 85 variables and commands for the HP and 134 for the Canon). The two we chose have several different main functions, including copying, faxing, scanning, and photo manipulation, that all must be represented in the user interface. They also have many special configuration options for each of the main functions, which make the initial setup process difficult and time-consuming.
- Two simple copier interfaces were used previously to demonstrate the features of Uniform [10], and we wanted to test Uniform with more realistic appliances with similar functionality.
- Although it was not possible for the PUC to actually control the all-in-one printers, simulating this control was easy to achieve by configuring a computer to print documents on the printers with the correct appearance based on the task the user was currently performing.

This resulted in a realistic setting for users of the PUC interfaces, which allows for better comparisons of the PUC interfaces with the actual appliance interfaces.

The existing manufacturers' interfaces from both printers were used for the comparisons conducted in the studies. The generated interfaces produced by the PUC system were presented on a Microsoft PocketPC device (see Figure 1).

This study represents the first quantitative evaluation of the PUC with real users. Previously, we have evaluated the completeness of the implementation of the PUC [8, 10]. We have also performed user studies of appliance interfaces on handheld devices, but these interfaces were not created automatically [11].

RELATED WORK

Research in interface generation has a long history dating back to some of the earliest User Interface Management Systems (UIMSs) developed in the mid-80's, such as COUSIN [5]. This work led to creation of systems in the late 80's and early 90's, such as UIDE [17], ITS [20], Jade [19], and Humanoid [18], which required designers to specify models of their applications. The automatically generated interfaces could generally be modified by a trained interface designer to produce a final user interface. These interfaces were sometimes called *model-based user interfaces* because of the models underlying their creation.

These early model-based systems had several drawbacks. Most notably, creating the models needed for generating an interface was a very abstract and time-consuming process. The modeling languages had a steep learning curve and often the time needed to create the models exceeded the time needed to program a user interface by hand. Finally, automatic generation of the user interface was a very difficult task and often resulted in low quality interfaces [7]. Most systems moved to designer-guided processes rather than continue using a fully automatic approach.

Two motivations suggested that continued research into model-based approaches might be beneficial:

Very large scale user interfaces assembled with existing techniques are difficult to implement and later modify, and detailed models of the user interface can help organize and partially automate the implementation process. The models can then be used to help designers re-visit the interface and make modifications for future versions. Mobi-D [15] and TERESA [6] are two notable approaches in this area.

A recent need for *device-independent interfaces* has also motivated new research in model-based user interfaces and specifically on fully automated generation. Work in this area has also begun to explore applications of automatic generation to create interfaces that would not be practical through other approaches. For example, the PUC's Uniform layer [10] generates interfaces that are personally consistent with each user's previous experience.

Three relevant automatic generation systems are Xweb [13], ICrafter [14], and the Ubiquitous Interactor [12]. Xweb enables users to interact with services through automatically generated interfaces in several modalities and client styles, including speech, desktop computers, and pen-based wall displays. ICrafter is designed to distribute interfaces for controlling services to any interactive device that wishes to display those interfaces. ICrafter's innovation is its ability to aggregate the user interfaces for multiple services together based on a set of programming interfaces which identify services that can be used together. The Ubiquitous Interactor also generates interfaces for services, but provides service provider's with the unique ability to supply hints about how the generated interface should appear and include brand marks and interactions.

Most automatic interface generation systems, including the PUC, use a rule-based approach to create user interfaces. SUPPLE [2] instead uses a numeric optimization algorithm to find the optimal choice and arrangement of controls based on a cost function. The developers of SUPPLE have experimented with including a number of different factors in this cost function. Common factors to all are the cost of navigation between any two controls and the cost of using a particular control for a function. Additional costs have been included based on the common tasks that a user performs [2], consistency between interfaces for the same application generated on different platforms [3], and the physical abilities of the user (for assistive technology) [4].

All of these systems automatically generate interfaces, but to our knowledge no user studies have been conducted to evaluate the resulting interfaces. The closest reported study is of SUPPLE [2], which asked subjects without any design training to produce interfaces for a presentation room control panel. The developers then showed that SUPPLE could generate similar versions of each interface by varying the task information provided to the interface generator. The interface used in this study had only a few simple functions however, and users' performance on the SUPPLE interfaces was not measured or compared with any other interfaces.

BACKGROUND: THE PUC SYSTEM

The PUC system generates interfaces from specifications of appliance functionality using a rule-based approach [8]. In the system, handheld devices and appliances communicate over wireless networks using a peer-to-peer approach. When the user wishes to control an appliance, her handheld device connects to the appliance, downloads a functional specification from that appliance, and then generates an interface. The user can then use that interface to both remotely control the appliance and receive feedback on the appliance's state. Currently, graphical interface generators using the PUC framework have been implemented for the PocketPC, Microsoft's Smartphone platform, and desktop computers. A speech interface generator was also implemented using Universal Speech Interface techniques [16]. The PUC specification language is designed to be easy-to-

a. HP Photosmart 2610 **b. Canon PIXMA MP780**

Figure 2. The all-in-one printers used in our studies, with a larger view of the built-in user interfaces.

use, concise, and contain the information most important for generating user interfaces.

The PUC system is able to control real appliances, and adapters have been created to connect the PUC to an Audiophase stereo, a Sony camcorder, a UPnP camera, and several lighting systems. To test the completeness of the PUC appliance specification language, specifications have been written for many other appliances that could not be controlled directly. Over 30 different specifications have been written for appliances as diverse as VCRs, desktop applications like PowerPoint, a car navigation system, and an elevator. Simulators have been built for some of the appliances that could not be directly controlled, and a generic simulator has been built which enables Wizard-of-Oz-style simulation for the remaining specifications.

Recently, the PUC system has been augmented with a new feature called Uniform [10]. Uniform adds additional rules to the PUC that ensure *personal consistency*, which means that new interfaces are generated to be consistent with interfaces the user has seen in the past. While these algorithms ensure consistency, they also preserve the usability of any unique functions of the new appliance. This choice may affect the consistency of the generated interface in some cases, such as when the new appliance has a similar function that is more complex than the previous appliance. In this case, the complex functionality will be preserved, but the function may be moved, within the interface's structure, to a location similar to the previous appliance. The completeness of Uniform was tested qualitatively with two copiers and several complex VCRs, but the study reported here is the first to evaluate Uniform's effectiveness.

USER STUDY OF AUTOMATIC GENERATION

We start with a description of the interfaces and the procedure used in the study, followed by a presentation of the results and some discussion.

Interfaces

Six different interfaces were used in the study, which includes three different types for each of the two printers:

- **Built-in** interfaces on the two existing all-in-one printers (see Figure 2).
- **PUC** interfaces generated normally, i.e. without consistency, for the two printers (see Figure 1a-b).
- **Uniform** interfaces generated for one printer by the PUC's Uniform layer to be consistent with a PUC interface for the other printer (see Figure 1c-d).

PUC specifications for both all-in-one printers were needed by the PUC and Uniform to generate interfaces. The first author wrote the initial specification for the Canon printer and the second author wrote the initial specification for the HP printer. Different writers were used for the two specifications so that they would contain similarities and differences that might be found in a realistic scenario where the specifications were written by different manufacturers.

Both authors are experienced with writing specifications, and the initial drafts were produced in 2-3 days. The specifications were written using an approach that we would expect actual specification writers to take. We were generally faithful to the design of the actual appliances, and followed the structure and naming used in the manufacturers' manuals. We also took advantage of the features of the PUC specification language. For example, the language allows for multiple labels for each function and we added extra labels with further detail where appropriate. The PUC language also calls for authors to include as much organizational detail as possible in order to support generation on devices with small screens, and we followed this guideline. Debugging the specifications, such as ensuring that every variable had at least one label, took another 2-3 days. Note that this testing is similar to debugging a program or itera-

tively testing a user interface, but the advantage of the PUC system is that these improvements are only needed once and will migrate properly to interfaces generated on any platform.

An important point is that both specifications included all of the features of their appliances, even the features not tested. Therefore, the resulting generated user interfaces are *complete* in that they represent all of the features that could be accessed from the appliance's own user interfaces. The specification for the HP consists of 1924 lines of XML containing 85 variables and commands, and the specification for the Canon is 2949 lines of XML containing 134 variables and commands.

The PUC's Uniform layer also needs information about the similarities between specifications [10]. An automatic system was used to generate an initial set of mappings between the two all-in-one printer specifications. The first author then revised the resulting mappings to produce the complete set used in our study.

Protocol

Our protocol has three steps. Subjects first perform a block of eight tasks using an interface for one of the printers without any assistance. Next the subjects are taught the quickest way to perform the same tasks using the same interface and are not allowed to continue until they can complete every task correctly. Finally, subjects perform another block of the same eight tasks using an interface for the other printer, again without any assistance. This design allows us to measure performance without any previous experience (the first block of tasks) and to compare the effect on performance of any knowledge transfer from the first interface to the second. The instruction period between the task blocks simulates expertise on the first interface, as if the users were very familiar with that appliance, to maximize the effects of knowledge transfer.

Before starting each task, subjects were seated such that they were facing away from the interface. First, the subject was given the instructions for that task and allowed to read them. Once they were comfortable with the instructions they were allowed to turn around and begin the task. Time was recorded from the moment the subject turned around to the moment the final step of the task was completed. Subjects were allotted a maximum of 5 minutes to perform each task and were not allowed to move on until they succeeded or the maximum time had passed. We chose 5 minutes based on our pilot studies, which suggested that most subjects would finish within that window or else would never succeed.

We manipulated the order in which subjects saw the printers and the particular interfaces that they saw. Three different configurations of interface type were used:

- **Built-in:** One built-in interface followed by the other built-in interface (e.g. HP built-in followed by Canon built-in or vice versa).
- **AutoGen:** PUC interface for one appliance followed by the PUC interface for the other (e.g., HP PUC followed by Canon PUC or vice versa).
- **Consistent AutoGen:** PUC interface for one appliance followed by the Uniform interface for the other appliance generated to be consistent with the first interface (e.g., HP PUC followed by Canon Uniform generated to be consistent with HP PUC).

The Consistent AutoGen configuration is designed to fulfill the assumption of the Uniform's consistency algorithms, which is that users will be able to transfer knowledge when they encounter a new device because they are familiar with a previous interface.

These three configurations allow us to test both usability and consistency. Usability is tested by comparing the Built-in configuration with the other two. Consistency is tested by comparing the AutoGen and Consistent AutoGen configurations. All configurations were tested with both of the possible orderings (HP followed by Canon and vice versa), and subjects were appropriately counter-balanced into each of the resulting six groups.

Tasks

We chose eight tasks for subjects to perform during the study. The tasks were chosen to be realistic for an all-in-one printer, cover a wide range of difficulties, and be as independent from each other as possible (so success or failure on one task would not affect subsequent tasks). The last point was especially important, because we wanted to minimize the possibility that a subject might notice an element used in a future task while working on an earlier task. We also tried to minimize this effect by presenting the next task description only after subjects had completed their previous task. This does not prevent subjects working on their second block from remembering the tasks from the first block and the instructional phase.

The tasks we used, in the order they were always presented to subjects, are listed below. We chose not to vary the order of tasks for each subject so that whatever learning effects might exist between the tasks, despite our best efforts to eliminate such effects, would be the same for each subject. The task wording is paraphrased for brevity:

1. Send a fax to the number stored in the third speed dial.
2. Configure the fax function so that it will always redial a number that is busy.
3. Configure the fax function so that any document received that is larger than the default paper size will be resized to fit the default.
4. Configure the fax function so that it will only print out an error report when it has a problem receiving a fax, and not when it has a problem sending.
5. Make two black-and-white copies of the document that has already been placed on the document scanner.

6. Imagine you find the copies too dark. Improve this by changing one setting of the device.
7. Given a page with one picture, produce one page with several instances of the same picture repeated.
8. The device remembers the current date and time. Determine where in the interface these values can be changed (but changing them is not required).

We were careful not to use language that favored any of the user interfaces being tested. In some cases this was easy because all interfaces used the same terminology. In other cases we used words that did not appear in any of the interfaces. We also used example documents, rather than language, to demonstrate the goal of task 7.

Participants

Forty-eight subjects, twenty-eight male and twenty female, volunteered for the study through a centralized sign-up web page managed by our organization. Most subjects were students at the local universities and had an average age of 25 and a median age of 23. We also had 3 subjects older than 40 years. Subjects were paid $15 for their time, which varied from about 40 minutes to 1.5 hours depending on the configuration of interfaces being used. Subjects were randomly assigned to conditions.

Evaluation

To evaluate the performance time data, we performed a Mixed Model analysis using log(time) as the dependent variable. The log of time was used to make the distribution of our time data more normal for analysis. Interface Type (Built-in, PUC, or Uniform), Appliance (HP or Canon), Block # (1-2), and Task # (1-8) were modeled as fixed effects, and Subject # was modeled as a random effect. No interactions were included in the model reported here, though we tried models with all combinations of two degree interactions and found none of them to be significant.

Our analysis showed that all of the fixed effects had a significant effect on performance time. Most interesting is the effect of Interface Type ($F_{1,99} = 49.46$, $p < 0.001$). A Tukey HSD post-hoc test found each of the interface types to be significantly different from the others: PUC was faster than Built-In ($t(49) = 7.95$, $p < 0.001$), Uniform was faster than Built-In ($t(89) = 9.65$, $p < 0.001$), and Uniform was faster than PUC ($t(581) = 3.86$, $p < 0.001$). Least squares means for performance on each were (converted from log space; all units are seconds): $M_{built\text{-}in} = 64.48$, $M_{puc} = 27.48$, $M_{uniform} = 18.31$.

Task also had a significant effect on performance time ($F_{1,709} = 77.65$, $p < 0.001$). A Tukey HSD post-hoc test identified 4 overlapping groups of tasks for which users had significantly different performance. The most difficult group contained tasks 1, 2, 3, and 7, and the next slightly less difficult group contains tasks 1, 2, 3, and 4. This indicates that task 7 was more difficult than task 4 ($t(709) = 3.58$, $p < 0.001$), with tasks 1, 2, 3 somewhere between the two. The third group contains tasks 6 and 8, and was significantly less difficult than the previous two groups ($t(709) = 17.74$, $p < 0.001$). The fourth, least difficult, group contains only task 5, which was found to be significantly easier than all of the other tasks ($t(709) = 14.61$, $p < 0.001$).

Appliance also had a significant effect on performance time ($F_{1,709} = 50.62$, $p < 0.001$), with the Canon appliance being more difficult than the HP appliance. This is consistent with our experiences with the printers and our observations of the subjects. The effect of Block was also significant ($F_{1,756} = 21.21$, $p < 0.001$). This indicates that subjects' speed increased over the course of the experiment as they became more familiar with the tasks or the general concept of an all-in-one printer.

An important goal of our design was to limit task failures to between 5-10%, which guided our choice of 5 minutes as the maximum task time. Over all 768 tasks performed by our subjects, 57 failures occurred giving a 7.5% failure rate, which is within our goal range.

To evaluate task failures, we performed a Mixed Model analysis with failures per block as the dependent variable. Unlike the model for performance time where task # was an independent variable, for this model failures across all eight tasks were aggregated because the total number of failures was very small. In this analysis, Interface Type, Appliance, and Block # were included as fixed effects and Subject # as a random effect. The interaction of Interface Type with Appliance was also found to be significant and therefore was also included in the model as a fixed effect. No other interactions were found to be significant, and thus were not included in the model reported here. All averages stated as part of this analysis are least squares means.

The analysis found all of the fixed effects to be significant. Significantly more failures occurred in Block #1 ($F_{1,49.73} = 4.21$, $p < 0.05$), with an average of 0.79 failures per block across all tasks in Block #1 as compared to an average of 0.42 failures per block across all tasks in Block #2. There were also significantly more failures with the Canon printer as compared to the HP ($F_{1,55.53} = 11.58$, $p < 0.002$), with an average of 0.92 failures per block for the Canon as compared to an average of 0.29 failures per block for the HP.

The interaction of Interface Type and Appliance was also significant ($F_{1,62.04} = 4.29$, $p < 0.02$), seemingly because there were significantly more failures with the Built-In Canon interface as compared to any of the others (as shown by a Tukey HSD post-hoc test, $t(85.17) = 5.86$, $p < 0.001$). There were an average of 1.94 failures per block using the Built-In Canon interface as compared to 0.63 average failures per block for the Built-In HP interface and even less for the other interface and appliance combinations.

There was also a significant difference in failures due to Interface Type ($F_{1,65.97} = 7.53$, $p < 0.002$). A Tukey HSD post-hoc test showed that there was a significant difference ($t(50.75) = 3.69$, $p < 0.001$) between the Built-In interfaces

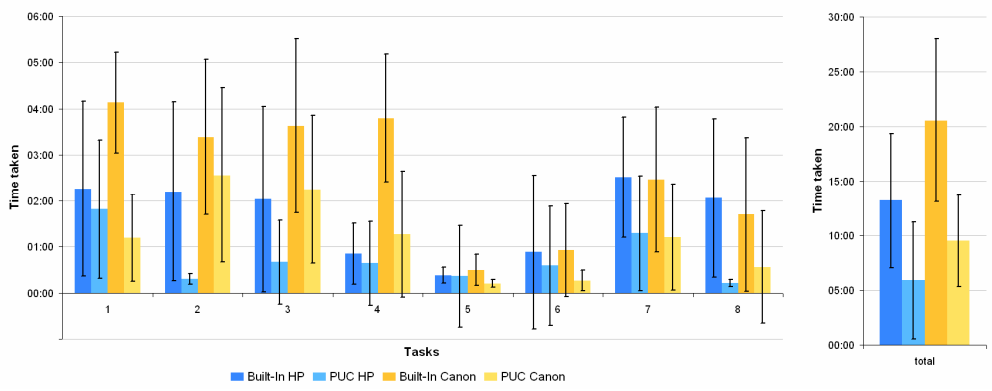

Figure 3. Results of the first block of tasks, showing the Built-In configuration compared with the other two for each appliance.

			Tasks 1	2	3	4	5	6	7	8	Total
Time	HP	Built-In	02:16	02:12	02:02	00:51	00:23	00:53	02:31	02:04	*13:12*
		PUC	01:49	00:18	00:40	00:39	00:22	00:35	01:18	00:13	*05:54*
	Canon	Built-In	04:08	03:23	03:38	03:48	00:30	00:56	02:28	01:42	*20:33*
		PUC	01:12	02:34	02:15	01:17	00:12	00:16	01:13	00:34	*09:32*
Failures	HP	Built-In	2	2	2	0	0	1	1	1	*9*
		PUC	2	0	0	0	0	0	0	0	*2*
	Canon	Built-In	3	3	5	3	0	0	1	1	*16*
		PUC	0	5	2	1	0	0	1	1	*10*

Table 1. Average completion time and total failure data for the first block of tasks. The PUC rows combine data from both the AutoGen and Consistent AutoGen interface configurations. N = 8 for Built-In and N = 16 for PUC.

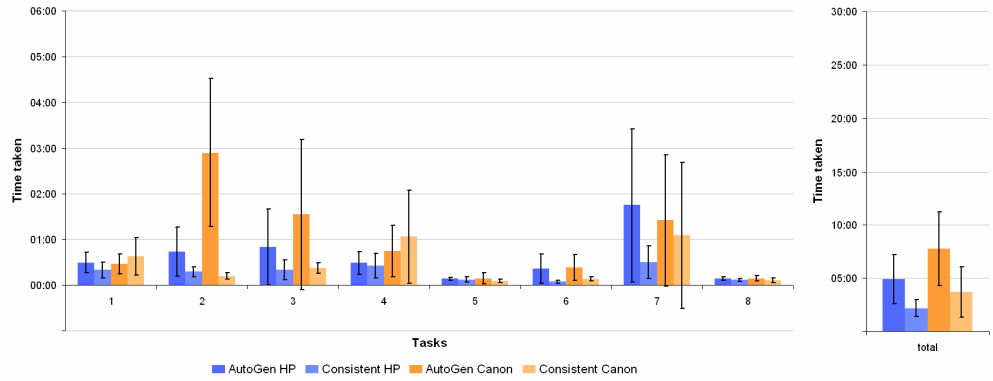

Figure 4. Results of the second block of tasks, showing the AutoGen configuration compared to the Consistent AutoGen configuration for each appliance.

			Tasks 1	2	3	4	5	6	7	8	Total
Time	HP	AutoGen	00:29	00:43	00:50	00:29	00:08	00:22	01:45	00:08	*04:54*
		Consistent	00:20	00:17	00:20	00:25	00:07	00:04	00:30	00:07	*02:10*
		Built-In	01:38	01:23	00:37	00:39	00:18	00:16	03:19	00:45	*08:55*
	Canon	AutoGen	00:28	02:54	01:33	00:44	00:09	00:23	01:25	00:09	*07:45*
		Consistent	00:38	00:12	00:22	01:03	00:05	00:08	01:05	00:06	*03:39*
		Built-In	03:15	02:24	02:42	02:14	00:11	01:42	02:42	00:35	*15:44*
Failures	HP	AutoGen	0	0	0	0	0	0	0	0	*0*
		Consistent	0	0	0	0	0	0	0	0	*0*
		Built-In	0	0	0	0	0	0	1	0	*1*
	Canon	AutoGen	0	2	1	0	0	0	0	0	*3*
		Consistent	0	0	0	0	0	0	1	0	*1*
		Built-In	4	2	3	2	0	2	2	0	*15*

Table 2. Average completion time and total failure data for the second block of tasks. N = 8 for all three interface configurations.

and the others, but there was no significant difference between the PUC and Uniform interfaces ($t(67.26) = 0.223$, $p = 0.82$). There was an average of 1.28 failures per block with the Built-In interfaces as compared to less than 0.3 failures per block for the other two interface types.

Figure 3 shows the performance times for the first block of tasks. Because the same interfaces were used in the first block of both the AutoGen and Consistent AutoGen interface configurations, the results of these two configurations have been combined into the PUC category. Table 1 shows the same performance times, along with the total number of failures that occurred in the first block of tasks. This figure and table should be used primarily to evaluate the usability of the PUC system.

Figure 4 shows the performance times for the second block of tasks for the AutoGen and Consistent AutoGen interface configurations. Table 2 shows the performance times and failures for the second block of tasks for all three interface configurations. This figure and table should be used primarily to evaluate the effect of consistency between each of the different interfaces.

Discussion of Usability

The results show that users perform faster using the PUC and Uniform interfaces as compared to the printers' built-in interfaces. The benefits seem to apply across all of the tasks (see Figure 3), though the extent of the benefit varies depending on the task and the appliance. There does not appear to be a trend with respect to task difficulty. Tasks 1-4 and 7 were found to be the most difficult in the analysis, but the performance improvement does not seem to be different overall for these tasks than for the other easier tasks.

Task 2 had the most failures by users of the PUC interfaces by a wide margin, and all of these failures occurred on the Canon interface. We believe task 2 was particularly hard because the Canon printer has many configuration features for sending and receiving faxes, which are complex, overlap, and use language that is difficult to understand. These functions were difficult to represent cleanly in the PUC specification language, which may have carried their complexity through to the generated interfaces.

This study, at least for the first block of tasks, compares the performance of novice users. There is then a question of whether the PUC would be equally successful for expert users. As users become experts, they are less likely to make mistakes, which would probably benefit the harder-to-use Built-In appliance interfaces more than the PUC interfaces. However, fewer steps are required to navigate to and use most functions in the PUC interfaces so there is less for users to learn. Furthermore, the PUC interfaces provide more visual context for the user's current location in the interface. We believe that these features would allow users to become experts with the PUC interface faster than the Built-In interfaces. Unfortunately, this cannot be determined from the data collected in this study.

Discussion of Consistency

The results show that users perform faster using the Uniform interfaces as compared to the PUC interfaces. Much of the benefit from consistency for both appliances seems to be due to four tasks: 2, 3, 6, and 7. This was expected, because the normal PUC interfaces for these appliances were already consistent for tasks 1 and 8, and thus did not benefit from any change in the consistent interfaces. We had hoped to see consistency effects for the remaining tasks, but other factors seem to have affected tasks 4 and 5.

The difference between the two PUC printer interfaces for task 5 (copying) involves the placement of the copy and cancel buttons on the screen (see Figure 1). Uniform was able to change its interfaces to make the button placements consistent with the previous interface that the subject had seen. Despite the change, there was not an appreciable decrease in task time. Because the buttons were prominent in both the PUC and Uniform interfaces, it may be that the visual search was so quick for subjects to perform that little performance benefit was gained from remembering the previous location of the button.

Uniform was able to make one change to ensure consistency for task 4 (changing the fax error printing). The function needed for this task is located with other fax configuration functions that appear in different places on the two printers: in the fax mode on the HP and in the setup section of the Canon interface. The change for consistency performed by Uniform is to move all the configuration functions to the location where the user originally saw them. From observations of subjects' actions, it appeared that this manipulation worked in the studies. Unfortunately, the error reporting function was also different between the two appliances in a way that Uniform could not manipulate. When using the HP interface made to be consistent with the Canon interface, users needed time to understand how the functions were different before they could make the correct change. When using the Canon interface consistent with the HP interface, the interface generator made the unfortunate choice of placing the needed functions in a dialog box accessible by pressing a button. The button to open the dialog was placed next to several other buttons, which distracted subjects from the button they needed to find.

It is important to note that there are no situations where the PUC's consistency algorithms make the interface worse for users, even for task 4 on the Canon interface generated to be consistent with the HP. Uniform is able to provide benefits when there are similarities between the appliances and it does not hurt the user when there are differences.

A question to ask is whether the apparent benefits of consistency could be due to some other factor in the generation process. We do not believe this is likely, because the rules added by Uniform only make changes to the new interface based on differences with a previous interface. These rules do not perform other modifications that might improve the user interface independent of consistency.

GENERAL DISCUSSION

This study has shown that the PUC can improve usability by moving appliance interfaces to another platform with improved interaction primitives. Using automatic generation to create appliance interfaces offers flexibility in the design of the interface, which allows interfaces to be modified for each particular user. The consistency feature that we studied here is one example, and our study showed that consistency can be beneficial to users.

An important question is: what allows the PUC to generate interfaces that are better than the built-in interfaces on the appliances? In part, the PUC benefits from moving interfaces to a device with better interactive capabilities, but the PUC also uses good HCI practice in its generation, which avoids some problems often seen in appliance interfaces. First, all buttons, sliders, etc. presented in PUC interfaces are used for only one function. In contrast, most appliances overlap multiple functions on their buttons. For example, both printer interfaces provide a number of multi-purpose buttons, including directional pads, ok buttons, and number pads (see Figure 2), whose behavior changes depending upon the function selected through the printer's menu. This was a particular problem for the built-in Canon interface, which has many modes where certain buttons cannot be used and for which there is no feedback. Users must experiment to determine which buttons can be pressed in which situations. The PUC provides feedback by graying-out controls that are not currently available.

An important issue with this study is the fairness of the comparison for usability. Why not instead compare the automatically generated interfaces with hand-designed appliance interfaces provided on a similar PDA? After all, some portion of the usability improvement enjoyed by the PUC interfaces is likely due to the improved interaction primitives and larger screen afforded by the PocketPC device on which the interfaces were shown. While we could do this comparison, it would not be realistically addressing the limitations faced by today's consumer electronics manufacturers. Costs from both development and manufacturing must be heavily minimized in that industry in order to produce profitable products, and it is simply not economical to include the user interface hardware from a PDA on every appliance or to hire a team of usability experts to spend months carefully designing an interface [1].

How then does the PUC address these limitations? First of all, it makes use of a device that users already have or are likely to get in the near future: a mobile phone or PDA. In the PUC model, this device would be used across all appliances, so it is reasonable to expect that consumers might be willing to spend a little more on their device to have a better experience with all of their appliances. With the increasing popularity of high-performance smart phones, such as the Apple iPhone, many more people will be carrying devices capable of PUC-like features. Second, the cost of adding PUC functionality to an appliance is relatively small in both development and manufacturing and might be offset by reducing the existing interface on the appliance.

The main development cost is in writing a specification for the appliance. A design goal of the PUC specification language was conciseness and ease-of-use. As discussed earlier, the specifications used in this paper were each written in 2-3 days by experienced authors starting with no knowledge of the appliance. Appliance developers may not be experienced specification authors, but they will be experienced with the appliance. In previous work, we have shown that the language is easy to learn: subjects with no knowledge of the language were able to learn it in about 1.5 hours and produce a medium-sized specification in about 6 hours [10]. Manufacturers can also benefit if they have already written a functional specification of their appliance for internal purposes, which is quite common. Although this specification would not be in the same format and might not have all of the same details as a PUC specification, it would provide a good starting point. Overall, we estimate that a new manufacturer might require two weeks to write and debug a new specification (twice the amount of time it took us), which is likely to be substantially faster than designing, implementing and testing a new graphical interface.

If it is necessary to add a wireless radio transceiver to support Bluetooth or another wireless network, then manufacturing costs for each appliance could slightly increase with the PUC technology. There might also be an additional cost for a more powerful microcontroller, but we believe that the processing power already in most appliances should be able to manage the PUC communication protocol in addition to the appliance's main function. It is important to note that these extra costs could be compensated for by removing some elements from the existing appliance interface. For example, a manufacturer might remove the buttons for some complex functions in favor of only offering these through a PUC interface. If the PUC became more widespread, it might be possible to eliminate most of the on-appliance interface and also any remote control that might ship with the appliance.

The question remains whether manufacturers would be willing to adopt a technology like the PUC. Although usability is becoming more of a marketing point, it is still not clear that consumers value it over price except in a few instances (e.g. the iPod). We believe usability is a growing desire among consumers however, which may lead manufacturers towards this kind of technology in the future.

Another reason to adopt technology like the PUC could be for its other benefits, such as improved flexibility for the user interface. Consistency is one such benefit, though users may be more in favor of it than manufacturers. In particular, manufacturers may object to consistency because branding may be removed from interfaces, and, worse still, branding from a competitor may be added in its place. Our position is that branding which affects the usability of an appliance, such as custom labels for certain functions or particular sets

of steps needed to complete particular tasks, is not good for the user and the consistency system should be allowed to modify them. However, branding marks, such as company names, logos, etc., should be preserved appropriately. Support for branding marks and consistency of those marks is a feature that may be added to the PUC system in the future.

The study presented here does have some limitations. We used only one type of appliance, the all-in-one printer, and only tested two instances of this type. As discussed earlier, we believe that the printers we chose are representative of most complex appliances. They also required the use of many of the PUC specification language's advanced features, such as lists and Smart Templates [9]. Although we only used two printers, we carefully chose both to be complex and representative of different common interface styles. We also chose the HP in part because it had, in our estimation, a better interface than most other all-in-one printers.

The results of our study show that the PUC can generate usable printer interfaces, but what about for other kinds of user interfaces? It would probably be beneficial to have features, like personal consistency, built into all of our user interfaces. Currently, the PUC cannot generate interfaces that use direct manipulation, such as a painting application, and the PUC may have difficulty generating interfaces for large amounts of structured data, such as a calendaring system. It might also be possible for the PUC's consistency algorithms to be applied in other systems. Others [2] have automatically generated interfaces for ubiquitous computing applications, and it might be possible for Uniform to be integrated into these systems. It might also be possible to apply Uniform to hand-designed interfaces, provided that a model of the interface was available to guide the interface modifications. Developing systems that automatically modify user interfaces is a promising direction for future work.

CONCLUSION

The results of the study in this paper show that the PUC can automatically generate interfaces which are more usable and provide personal consistency. This suggests two implications for future user interface design and research. For design, it suggests that automated processes should be considered in products where interfaces may be constrained by external factors or individual user customization may provide substantial benefits. For research, it suggests that an important direction for future work is developing new techniques that use automatic generation to create interfaces that are customized for each individual.

ACKNOWLEDGMENTS

We would like to thank the reviewers and Daniel Avrahami for providing insightful feedback that helped us improve the paper substantially. This work was funded in part by grants from the Pittsburgh Digital Greenhouse, Microsoft, General Motors, and under Grant No. IIS-0534349 from the National Science Foundation. 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., Bennett, R.W., Endo, T., van Nes, F.L., Strubbe, H.J., and Gentner, D.R. Interfaces for consumer products: "how to camouflage the computer?" in *CHI'1992*: 287-290.
2. Gajos, K., Weld, D. SUPPLE: Automatically Generating User Interfaces, in *Intelligent User Interfaces*. 2004: 93-100.
3. Gajos, K., Wu, A., and Weld, D.S. Cross-Device Consistency in Automatically Generated User Interfaces, in *2nd Workshop on Multi-User and Ubiquitous User Interfaces*. 2005: 7-8.
4. Gajos, K.Z., Long, J.J., and Weld, D.S. Automatically Generating Custom User Interfaces for Users With Physical Disabilities, in *ASSETS*. 2006: 243-244.
5. Hayes, P.J., Szekely, P.A., and Lerner, R.A. Design Alternatives for User Interface Management Systems Based on Experience with COUSIN, in *SIGCHI'85*: 169-175.
6. Mori, G., Paterno, F., and Santoro, C., Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Transactions on Software Engineering*, 2004. 30(8): 1-14.
7. Myers, B.A., Hudson, S.E., and Pausch, R., Past, Present and Future of User Interface Software Tools. *ACM Transactions on Computer Human Interaction*, 2000. 7(1): 3-28.
8. Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., and Pignol, M. Generating Remote Control Interfaces for Complex Appliances, in *UIST'2002*: 161-170.
9. Nichols, J., Myers, B.A., and Litwack, K. Improving Automatic Interface Generation with Smart Templates, in *Intelligent User Interfaces*. 2004: 286-288.
10. Nichols, J., Myers, B.A., and Rothrock, B. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces, in *CHI'2006*: 611-620.
11. Nichols, J., Myers, B.A. Studying The Use Of Handhelds to Control Smart Appliances, in *23rd International Conference on Distributed Computing Systems Workshops (ICDCS)*. 2003: 274-279.
12. Nylander, S., Bylund, M., and Waern, A. The Ubiquitous Interactor - Device Independent Access to Mobile Services, in *Computer-Aided Design of User Interfaces (CADUI)*. 2004: 271-282.
13. Olsen Jr., D.R., Jefferies, S., Nielsen, T., Moyes, W., and Fredrickson, P. Cross-modal Interaction using Xweb, in *UIST'00*: 191-200.
14. Ponnekanti, S.R., Lee, B., Fox, A., Hanrahan, P., and T.Winograd. ICrafter: A service framework for ubiquitous computing environments, in *UBICOMP 2001*: 56-75.
15. Puerta, A.R., A Model-Based Interface Development Environment. IEEE Software, 1997. 14(4): 41-47.
16. Rosenfeld, R., Olsen, D., and Rudnicky, A., Universal Speech Interfaces. *ACM interactions*, 2001. VIII(6): 34-44.
17. Sukaviriya, P., Foley, J.D., and Griffith, T. A Second Generation User Interface Design Environment: The Model and The Runtime Architecture, in *INTERCHI'93*: 375-382.
18. Szekely, P., Luo, P., and Neches, R. Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design, in *SIGCHI'92*: 507-515.
19. Vander Zanden, B. and Myers, B.A. Automatic, Look-and-Feel Independent Dialog Creation for Graphical User Interfaces, in *SIGCHI'90*: 27-34.
20. Wiecha, C., Bennett, W., Boies, S., Gould, J., and Greene, S., ITS: A Tool for Rapidly Developing Interactive Applications. *ACM Transactions on Information Systems*, 1990. 8(3): 204-236.