Demonstrating the Viability of Automatically Generated User Interfaces

ABSTRACT

We conducted two studies that demonstrate automatically generated interfaces can be more usable than interfaces created by human designers. The first study shows that users of automatically generated interfaces for two all-in-one printers were twice as fast and four times more successful at completing tasks than users of the manufacturer's original interfaces. The second study shows that algorithms for automatically generating *consistent* user interfaces can provide additional benefits by allowing users to perform tasks with a new interface twice as fast as users of an interface generated without consideration for consistency. These two studies 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 this 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].

Submitted for Publication

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 user may benefit if his or her interfaces are specifically designed 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 address the challenges of previous systems. In particular, Myers et al. [7] said that previous systems were not adopted in part because "generated interfaces were generally not as good as those that could be created with conventional programming techniques" [7].

In this paper, we present two user studies that examined the usability of interfaces automatically generated by the Personal Universal Controller (PUC) system [8] (see Figure 1). The first study examined the usability of the generated in-

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.

terfaces compared to existing human-designed interfaces for the same functionality, with the hypothesis that interface quality is no longer a limiting factor for automatically generated interfaces. The results were that users of the automatically generated interfaces were *twice as fast* and *four times more successful* than users of the existing interfaces for a set of eight independent tasks with varying difficulty.

The second study examined the PUC's Uniform layer, which automatically generates interfaces that are consistent with the user's previous experience [10]. Our hypothesis was that automatically generated interfaces can provide benefits beyond those shown in the first study through user customizations that would be impractical for human designers to provide. In our study, we first trained users on the same eight tasks from the first study using one interface. After users could successfully perform these tasks, we asked them to perform the same tasks on a second different interface with similar functionality. We found that users are *twice as fast* when the second interface is generated by the PUC to be consistent with the first interface, as compared to when the second interface is generated with the consistency algorithms disabled.

Both user studies compare 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-printers available for home use at the time of their purchase. We chose the all-in-one printers as our appliances in these studies 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-printers were at least as complicated, if not more so, than many of the other appliance types that have been explored by the developers of the PUC system (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 PUC's consistency features [10], and we wanted to understand whether the PUC's consistency algorithms would work 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 paper begins with a brief review of the extensive research on automatically generating user interfaces, focusing on recent approaches examining how automatic generation can provide benefits that would not be practical for human designers to provide. We continue with an overview of the PUC system, followed by an in-depth description of our studies and their results.

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]. The original goal of these systems was to automate the design of the user interface so that programmers, who were typically not trained in interface design, could produce applications with high quality user interfaces. This work led to creation of systems in the late 80's and early 90's, such as UIDE [16], ITS [19], Jade [18], and Humanoid [17], which required designers to specify models of their applications that could then be used to automatically generate a user interface. The 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 manually 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 use 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 [14] 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 consistency feature [10] generates interfaces that are personally consistent with each user's previous experience.

Xweb [12] enables users to interact with services through several different modalities and client styles, including speech, desktop computers, and pen-based wall displays. The wide-range of interfaces is supported through the automatic generation of interfaces, which are built from an XML description of a service's capabilities.

ICrafter [13] is designed to distribute interfaces for controlling services to any interactive device that wishes to display those interfaces. As with Xweb, specifications of the services are written in XML and interfaces can be displayed on multiple devices in multiple modalities. 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 [11] also generates interfaces for services, but provides service provider's with the unique ability to supply hints about how the generated interface should appear. This gives the service providers control over the generated interfaces and allows them to 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 of their functions 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 interface design training to produce interfaces for a presentation room control panel. The developers then showed that SUPPLE could generate similar versions of each of these interfaces 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

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.

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 [15]. The PUC specification language is designed to be easy-to-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 previous 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. Uniform was tested qualitatively with two copiers and several complex VCRs, but no studies quantitatively evaluating user performance were conducted until the current paper.

STUDIES OF AUTOMATIC GENERATION

Our discussion of the two user studies starts with a description of the interfaces we compared and the common protocol used for both studies. This is followed by sections presenting and discussing the results for each of the studies.

Interfaces

The studies compare PUC-generated interfaces with the manufacturers' human-designed interfaces for the same appliances, and compare PUC-generated interfaces with and without consistency for the two different printers. The manufacturers' interfaces for the two all-one-printers used are shown in Figure 2.

PUC specifications of both all-in-one printers were needed in order for the PUC 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 because these specifications are used for the consistency user study. We wanted the specifications to contain similarities and differences that might be found in a realistic scenario where the specifications were written separately by different manufacturers.

The specifications were also written using an approach that we would expect actual specification writers to take. Writers were generally faithful to the design of the actual appliances, but 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 necessary. 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 our authors followed this guideline. The initial specifications were tested with the interface generators to ensure correctness and went through several iterations before they were deemed of high enough

quality to be used for the studies. Note that this testing is similar to debugging a program or iteratively testing a user interface and is necessary to ensure that no functions are forgotten, understandable labels are used, etc. The advantage of the PUC system is that these improvements are only needed once and will migrate properly to interfaces generated on any platform.

Note also 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 consistency algorithms also need information about the similarities between specifications [10]. An automatic systems 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 consistency study.

The two specifications and the mappings between them were then used by the PUC to produce the four different interfaces used in our studies: PUC HP without consistency, PUC Canon without consistency, PUC HP generated to be consistent with the PUC Canon interface, and PUC Canon generated to be consistent with the HP (see Figure 1). Combined with the built-in interfaces for the two printers, this results in the six total interfaces used in our studies.

Protocol

The subjects using the PUC interfaces first had a short tutorial on the interface of the PocketPC handheld device. This was necessary because the PUC's design assumes that users will be familiar with the device they are using, and the PocketPC has several interface quirks that can frustrate users who are not aware of them (e.g. the Ok button in dialog boxes is located in the title bar at the top of the screen). Since the intention of the PUC is to work on people's own personal devices, it is reasonable to expect that they will be familiar with the user interface of the device itself.

All subjects performed a block of eight tasks on one of the six interfaces just described. After completing all of the tasks, the subjects received instruction on the quickest method of performing each of the tasks they had just performed. After receiving instruction on a task, subjects were required to perform the task again until they did not make errors. Additional instruction was available for a task as needed by the subject. Once the instruction period was completed successfully, the subject performed a second block of the same eight tasks on a different interface for the other all-in-one printer.

During both task blocks, users were required to figure out how to perform each task on their own and were *not* provided with a user manual or any other instruction on how to use the printer interfaces. Users were allotted a maximum of 5 minutes to perform each task and were not allowed to move on to the next task until they succeeded or the maximum period was complete. We chose 5 minutes based on several pilot studies that suggested that most subjects would finish within that window or else would never succeed. We recorded the time that it took subjects to complete each task. If a subject did not finish within the allotted period, we recorded his or her completion time as 5 minutes and marked the task as not being completed.

Our protocol has two independent variables: the type of interfaces that a subject used and the order in which the subject used the two all-in-one appliances. Three different configurations of interface type were used in our studies:

- **Built-in:** One built-in interface followed by the other built-in interface (e.g. HP followed by Canon).
- **AutoGen:** PUC interface without consistency for one appliance (e.g., HP) followed by the PUC interface without consistency for the other (e.g., Canon).
- **Consistent AutoGen:** PUC interface without consistency for one appliance (e.g., HP) followed by the PUC interface for the other appliance (e.g., Canon) generated to be consistent with the first interface (e.g., HP).

The Consistent AutoGen configuration is designed to fulfill the assumption of the PUC's consistency algorithms, which assume that users will receive a benefit from consistency 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 either of the others. Consistency is tested by comparing the AutoGen and Consistent AutoGen configurations. To test each of these configurations with both of the possible orderings (HP followed by Canon and vice versa) we use a 3x2 between-subjects study design. A within-subjects design is not possible because we must carefully control learning to compare performance for both the usability and consistency studies.

Tasks

We chose eight tasks for subjects to perform during each block of 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; however, this does not prevent subjects working on their second block from remembering the tasks from the first block.

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 was 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.
5. Make two black-and-white copies of the document that has already been placed on the scanner of the all-in-one printer.
6. Imagine you find the copies too dark. Improve this by changing one setting of the device.
7. Given a page with a picture, determine how to 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 forty minutes to an hour and a half depending on the configuration of interfaces being used. Subjects were randomly assigned to conditions.

Evaluation of Usability

To evaluate the usability of the PUC interfaces, we compared the task completion times and failures for the Built-in condition with the other two conditions. We are primarily interested in the data from the first block in each condition because the second block is influenced differently in each condition by the subjects' experiences in the first block.

Results

Figure 3 shows the average completion time for each of the tasks on each appliance, comparing the Built-In condition with the other two conditions combined (which we will refer to as the PUC condition). Note that data from the AutoGen and Consistent AutoGen conditions can be combined here because the same interfaces are used in the first block of both conditions. To compare completion times and failures in the first block, we conducted several one-way analyses of variance (ANOVA). For all of these analyses, n=8 in the Built-In condition and n=16 in the PUC condition. Table 1 shows data in more detail with analyses comparing user performance for each task.

On the HP appliance, subjects were significantly faster for total task completion time using the PUC interface ($F_{1,22}$ = 12.11, $p < 0.002$), completing all of the tasks in less than half the time (M=5:54 for the PUC interface vs. M=13:12 for the built-in interface). Subjects also failed significantly less often using the PUC interface ($F_{1,22}$ = 5.69, $p < 0.03$), with a fifth as many failures using the PUC interface as compared to the built-in interface (2 total failures for all users vs. 9).

Subjects overall had more difficulty using the Canon interfaces as compared to the HP interfaces across all conditions ($F_{1,46}$ = 6.25, $p < 0.02$), but we still see the same significant benefits for the PUC interface over the built-in interface. Again, subjects were significantly faster using the PUC ($F_{1,22}$ = 21.88, $p < 0.001$), with average total completion times of 9:32 for the PUC interface and 20:33 for the built-in interface (again about half the time). Subjects also failed significantly less often using the PUC ($F_{1,22}$ = 6.57, $p < 0.02$), with 10 total failures for all users over all tasks using the PUC interface and 16 total failures using the built-in interface (about 1/3 fewer failures on average).

We also performed the same analyses comparing the Built-In condition and the combined PUC condition for the data from the second block of tasks. All of these analyses were significant and matched the results for the first block, except for the number of failures over all tasks for the HP printer. In this case there were too few failures to make this analysis possible: zero failures for all 16 subjects using a PUC HP interface and only one failure for the 8 subjects using the built-in HP interface.

Discussion of Usability

The results show that users perform faster over all eight tasks using the PUC interfaces as compared to the printers' built-in interfaces.

For the Canon printer, the PUC interfaces are significantly faster for nearly all of individual tasks: tasks 3 and 8 are marginally significant and only task 2, automatically redialing a busy number, was not found to be different at all.

Task 2 was also the task most failed by users of the PUC interfaces by a wide margin. We believe task 2 was particularly hard for users because the Canon printer has many

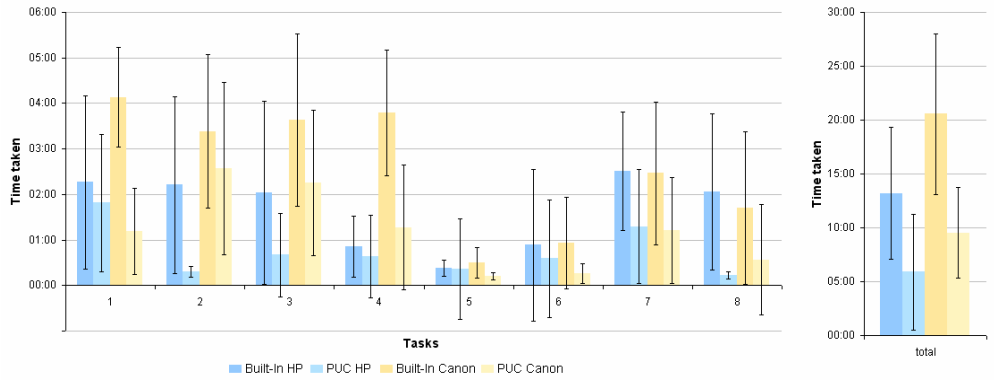

Figure 3. Results of the first block of tasks, showing the Built-In condition compared with the other two for each appliance.

			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 condition is the combination of the AutoGen and Consistent AutoGen conditions. N = 8 for the Built-In condition and N = 16 for the PUC condition. * indicates a significant difference between the Built-In and PUC conditions for that appliance ($p < 0.05$), and † indicates a marginally significant difference ($p < 0.1$). Completion times and total failures were compared with a one-way analysis variance and failures per task were compared with a one-tailed Fisher's Exact Test.

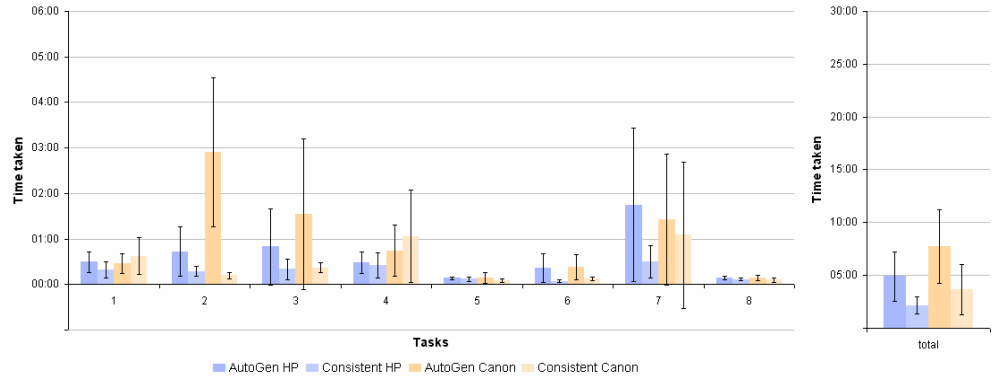

Figure 4. Results of the second block of tasks, showing the AutoGen condition compared to the Consistent AutoGen condition for each appliance.

			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 conditions. * indicates a significant difference between that row's condition and the Consistent AutoGen condition for that appliance ($p < 0.05$), and † indicates a marginally significant difference ($p < 0.1$). Completion times and total failures were compared with a one-way analysis variance and failures per task were compared with a one-tailed Fisher's Exact Test.

configuration features for sending and receiving faxes, which are complex, seemingly overlap with unrelated functions, and use language that is difficult to understand. These functions were difficult to represent cleanly in the PUC specification language and this may have carried their complexity through to the generated interfaces.

There are fewer individual tasks on the HP printer for which the PUC interface was significantly faster than the built-in interface: only tasks 2, 4, 7, and 8. We believe this is because the HP printer already has a well-designed interface and seemed to perform well, especially for the easier tasks. The tasks where the PUC interfaces excel are generally the more difficult tasks, like tasks 2 and 3, which require the users to find obscure settings deep in the interface.

We chose the five minute maximum completion time with a goal of limiting failures to between 5-10% of the total tasks. In this data there were 48 subjects performing 8 tasks each for 384 total tasks, and 37 failures were recorded. This gives a 9.6% failure rate, which is high but still within our goal range. Since the time measurements were cut off at 5 minutes, one might worry that this biased the results. However, more than 70% of the failures are found in the Built-In condition. This suggests that our results, which already show the Built-In condition to be slower overall, are likely to be correct since allowing more time would have only made that condition slower.

This study of usability, 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. 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, and the results of the second study suggest this may be true.

Evaluation of Consistency

To evaluate consistency, we compare the completion times of interfaces in the AutoGen and Consistent AutoGen conditions for the second block of tasks. We also compare the Built-In condition to the Consistent AutoGen condition, to see how consistency might further improve today's appliance interfaces.

Results

Figure 4 shows the average completion times for each task in the second block for the AutoGen and Consistent AutoGen conditions. Table 2 shows the same data in more detail and includes the Built-In condition and failure data for all the conditions. Again, we use one-way ANOVAs to compare the completion times of the various conditions. We do not discuss failures here because nearly all subjects were able to complete all their tasks in the AutoGen and Consistent AutoGen conditions (results of the analyses of failures are shown in Table 2).

On the HP appliance, subjects were significantly faster for total task completion time using the consistent PUC interface compared to the normal PUC interface ($F_{1,14} = 10.01$, $p < 0.007$) and the built-in interface ($F_{1,14} = 64.48$, $p < 0.001$). The total completion time for the consistent PUC interface was on average more than twice as fast as the normal PUC interface (M=2:10 vs. M=4:54) and more than four times faster than the built-in interface (M=2:10 vs. M=8:55).

Subjects were also significantly faster using the consistent PUC interface for the Canon printer, both compared with the normal PUC interface ($F_{1,14} = 7.60$, $p < 0.02$) and the built-in interface ($F_{1,14} = 16.89$, $p < 0.002$). The average total completion time for the consistent PUC interface was again more than twice as fast as the normal PUC interface (M=3:39 vs. M=7:45) and more than four times faster on average than the built-in interface (M=3:39 vs. M=15:44).

We also compared the total completion times for the two blocks of task for each of the three conditions. Neither the Built-In ($F_{1,30} = 3.24$, $p < 0.09$) or AutoGen ($F_{1,30} = 2.46$, $p < 0.14$) conditions were significantly different from the first block to the second, although the Built-In condition is marginally significant and the AutoGen condition may be trending in that direction. The Consistent AutoGen condition is significantly different from the first block to the second ($F_{1,30} = 10.45$, $p < 0.004$).

Discussion of Consistency

The results show that users perform faster over all eight tasks using the consistent interfaces as compared to either of the other interfaces. Much of this effect for both appliances is 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 change made to ensure consistency for task 5 (copying) involved changing the placement of the copy and cancel buttons on one screen (see Figure 1). Apparently the visual search for the new button placement did not affect subjects' speed compared to the normal PUC interfaces.

One change was made to ensure consistency for task 4 (changing the fax error printing). The function needed for this task is located with other fax configuration functions, which are located in different places on the two appliances: in the fax mode on the HP and in the setup section of the Canon interface. The change for consistency performed by the PUC 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 the PUC's consistency system 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.

For tasks 2 and 6 we see a significant benefit for consistency for both appliances. Tasks 3 and 7 both have a marginally significant benefit for consistency on just one appliance (task 3 on the HP and task 7 on the Canon). Similar to task 4, both tasks 3 and 7 are slightly different on the two appliances in ways that the PUC's consistency system cannot change. We believe this means that subjects were not able to leverage all of their previous knowledge and had to spend some of their time thinking about how the appliances worked, thus slowing them down.

It is important to note that there are no situations where the PUC's consistency algorithms make the interface significantly worse for users, even for task 4 on the Canon interface generated to be consistent with the HP. The consistency system 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 benefits that appear to be from consistency could be due to some other factor in the generation process. We do not believe this is likely, because the rules added for consistent interface generation only make changes to the new interface based on differences with a previous interface that the user has seen. These rules do not perform other modifications that might improve the user interface independent of consistency.

DISCUSSION

These two studies together have shown that the PUC can generate interfaces that exceed the usability of the manufacturers' own interfaces. Using automatic generation to create appliance interfaces allows 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 second study showed that consistency can be beneficial to users. Manufacturers may object to consistency however, 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.

An important question is: what allows the PUC to generate interfaces that are better than the built-in interfaces on the appliances? And what would be needed to improve the built-in interfaces? We believe PUC interfaces are better than the appliance interfaces for many reasons. First, the PUC does not use any overlapping controls. All buttons, sliders, etc. presented in a PUC interface 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 on their control panel, 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 manufacturer's interface on the Canon, which has many modes in which 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 addresses the feedback problem by graying-out controls that are not currently available.

The PUC's screen allows for longer and better labels to shown for each function. The screen also allows for a two-dimensional layout that can give clues to the organization of the interface. For example, the tab control allows users to see immediately that there are multiple groups of controls and what those groups are. Also, the functions displayed in the main portion of any given screen are grouped by functionality, which decreases the number of functions on any one screen and may make the interface easier to parse.

In order to improve the built-in interfaces to same usability as the PUC, manufacturers would probably need to invest in larger screens for their appliances. These screens would allow the organization of the interface to be clearer, and hopefully eliminate some of the need for multi-purpose buttons. Any physical buttons that are not always functional should have indicator lights that show when the button can be pressed. Many problems, such as poor labels, could be addressed with basic user-centered iterative design.

The question remains whether it is economical for manufacturers to make these improvements. Screens and indicator lights for buttons could add substantial manufacturing cost to an appliance that already has a low profit margin. 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 that the PUC could be an excellent solution for appliance manufacturers that currently find themselves in this situation.

The studies presented here do have some limitations. We used only one type of appliance, all-in-one printers, and only tested two instances of this type. As discussed earlier, we believe that the all-in-one printers we chose are representative of complex appliances as a whole. They also require the use of many of the PUC specification language's most advanced features, such as lists and Smart Templates [9]. Although we only used two all-in-one printers, we did

carefully choose them to both be complex and representative of different common interface styles. We also chose the HP in part because it had, in our estimation, the best interface of any all-in-one printer available.

Interface quality was a critical problem for previous automatic generation technologies, but there were other issues as well. In particular, two important problems were the need for developers to learn a new language for specifying the interface and the lack of predictability of the generated interfaces [7]. The PUC addresses both of these issues. Previously reported authoring studies have shown that the PUC specification language is easy to learn and use: new users with no previous knowledge were able to learn the language in about 1.5 hours and produce a specification for a low-cost VCR in about 6 hours [10]. Predictability is less of an issue for the PUC, because interfaces are generated for end users rather than designers. PUC interface generation is designed to be very stable however, meaning that the PUC will always generate the same output given the same inputs.

The results of our studies show that the PUC can generate usable appliance 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 could be used to generate any interface that does not use direct manipulation, such as a painting application, and does not involve a substantial amount of structured data, such as a calendaring system. The PUC has been used to generate partial interfaces for some desktop applications, such as PowerPoint. Others [2] have used their automatic generators to create interfaces for ubiquitous computing applications, and the consistency mechanism might be useful in these situations as well. Even applications for which an interface cannot be automatically generated, it could still take advantage of some of the benefits described here, provided that the application shipped with a model that could be used to help automatically modify the hand-designed interface for each user. Developing systems capable of automatically modifying user interfaces seems like a promising direction for future work.

CONCLUSION

The results of the two studies in this paper show that interfaces can be automatically generated which are more usable and provide personal consistency. This suggests two implications for the future of user interface design and research. For design, it suggests that automatic design should be considered in products where interfaces may be constrained by external factors or individual user customization may have 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 to each individual.

ACKNOWLEDGMENTS

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:Human factors in computing systems.* 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 *Proceedings of the 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: To appear.
5. Hayes, P.J., Szekely, P.A., and Lerner, R.A. Design Alternatives for User Interface Management Systems Based on Experience with COUSIN, in *Proceedings SIGCHI: Human Factors in Computing Systems.* 1985: 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 *Proceedings of CHI.* 2006: 611-620.
11. 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.
12. Olsen Jr., D.R., Jefferies, S., Nielsen, T., Moyes, W., and Fredrickson, P. Cross-modal Interaction using Xweb, in *Proceedings of UIST.* 2000: 191-200.
13. Ponnekanti, S.R., Lee, B., Fox, A., Hanrahan, P., and T.Winograd. ICrafter: A service framework for ubiquitous computing environments, in *UBICOMP 2001.* 2001: 56-75.
14. Puerta, A.R., A Model-Based Interface Development Environment. *IEEE Software*, 1997. **14**(4): 41-47.
15. Rosenfeld, R., Olsen, D., and Rudnicky, A., Universal Speech Interfaces. *ACM interactions*, 2001. **VIII**(6): 34-44.
16. Sukaviriya, P., Foley, J.D., and Griffith, T. A Second Generation User Interface Design Environment: The Model and The Runtime Architecture, in *Proceedings of INTERCHI.* 1993: 375-382.
17. Szekely, P., Luo, P., and Neches, R. Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design, in *Proceedings of SIGCHI.* 1992: 507-515.
18. Vander Zanden, B. and Myers, B.A. Automatic, Look-and-Feel Independent Dialog Creation for Graphical User Interfaces, in *Proceedings SIGCHI.* 1990: 27-34.
19. 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.

We present the first usability studies showing that automatically generated user interfaces can be superior to human-designed interfaces and provide additional benefits not practical to provide in human-designed interfaces.