Generating Consistent User Interfaces for Appliances

Jeffrey Nichols and Brad A. Myers
Human Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
{ jeffreyn, bam }@cs.cmu.edu

ABSTRACT
We are building a system called the *personal universal controller* (PUC) that automatically generates interfaces for handheld devices that allow users to remotely control all of the appliances in their surrounding environment. One of the goals of this system is to create interfaces that are consistent for the user. We are interested in two forms of consistency: with other interfaces on the same handheld device and with previously generated interfaces for similar appliances. While these problems differ slightly from the problem of ensuring an application has a consistent interface across multiple devices, we believe a solution to any of these problems will prove useful for solving the others. This paper discusses the challenges that we see for automatically generating consistent interfaces and ideas that we are pursuing to address the consistency problem.

Keywords
Automatic interface generation, consistency, Pebbles, appliances, personal digital assistants (PDAs), smart phones, personal universal controller (PUC)

INTRODUCTION
The goal of the *personal universal controller* (PUC) system [2] is to improve everyday appliance user interfaces by moving them from the appliance to a handheld device. We envision using PDAs and smart phones to control any computerized appliance in the office or home, such as stereos, microwave ovens, copiers, and answering machines. A key feature of the PUC system is that it automatically generates its user interfaces from an abstract description of the appliance. This allows our system to provide a number of benefits, including: interfaces can be personalized to the user, interfaces for multiple connected appliances can be combined together into one interface for the connected system, and generated interfaces can be made *consistent*.

PUC interfaces can be made consistent in three ways: with other interfaces on the user's handheld device, with interfaces the user has previously interacted with, and with other interfaces for the same appliance on different devices.

Submitted to the Second Workshop on Multi-User and Ubiquitous User Interfaces (MU3I) 2005

We have already addressed the first type of consistency by using standard interface toolkits and ensuring that our automatic generation rules conform to user interface guidelines for the device on which we are generating interfaces.

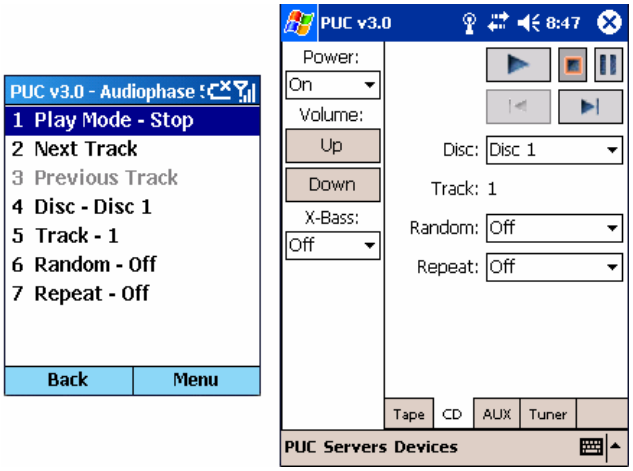
We are currently working to address the second consistency problem, as discussed below. We are addressing the third consistency problem by using similar generation rules on different platforms (see Figure 1), and by using familiar idioms, such as the conventional play and stop buttons for media players, with a technique we call Smart Templates [3]. We are focusing on the first two problems however, because we feel these types of consistency will contribute more to achieving high usability for our users.

CHALLENGES FOR CONSISTENCY
What does it mean to make an interface consistent? While there are design guidelines [4] that suggest some answers to these questions, many of these are not instructive for achieving previous interface or multi-device consistency.

A common mantra is to ensure that users can always find the functions they are looking for by always putting them in the same place. A key question is how to do this when interfaces are structured completely differently on different devices. For example, a PocketPC interface has a two-dimensional layout similar to a desktop interface, but a Microsoft Smartphone interface is list-based and navigated very differently from a standard desktop interface (see Figure 1). User studies are needed to evaluate how users map interface structures between different interface styles, and to determine whether consistency can be achieved for such different styles.

INTERFACE CONSISTENCY IN THE PUC SYSTEM
We have found that the problem of generating interfaces that are consistent with previous interfaces can be broken down into two sub-problems: finding previously generated interfaces that are relevant, and determining how to make the new interface consistent with those previous interfaces. We will only discuss the second sub-problem here.

Once previous interfaces with similar functions have been found, the interface generator can examine those interfaces and decide how to make the new interface consistent. The appropriate technique will depend on how similar the previous appliances are to the new appliance. There seem to be three levels of similarity, termed sparse, branch, and



a) b)

Figure 1. Two examples of interfaces generated by the PUC system for a) the Microsoft Smartphone and b) the PocketPC. Note that the interface for the Smartphone is list-based while the PocketPC interface uses a standard two dimensional layout. An interesting question: are these interfaces consistent?

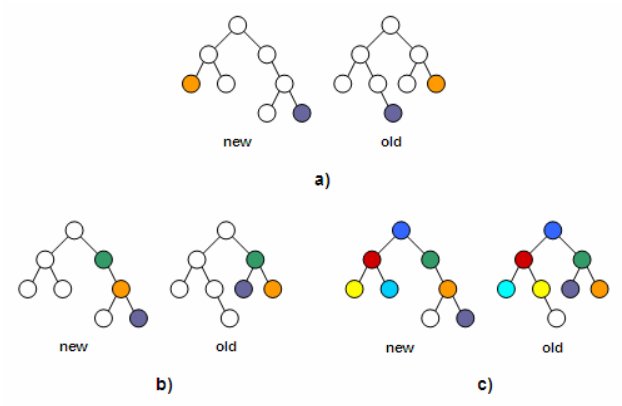


Figure 2. Examples of the three different levels of similarity, with trees representing the structure of the new and old interfaces and same shading indicating similar functions. a) *sparse similarity*: the appliances have a small number of similar functions spread throughout the tree. b) *branch similarity*: the appliances have a number of similar functions in one branch of the structure. c) *significant similarity*: the appliances share many similar functions, though they might be organized differently.

significant (see Figure 2), each of which suggests a different technique to achieve consistency. Appliances with sparse similarity will try to represent each similar function with the same interface controls that the user saw in the older interface. Appliances with branch similarity will try to integrate into the new interface the layout and organization of the related functions in the previous interface. Appliances with significant similarity will try to replicate the same layout and organization in the new interface that the user has seen in previous interfaces.

One of the difficulties with the significant similarity case is deciding how to deal with the few functions that are not shared across appliances. An important question to answer here is the importance of visual consistency. If visual consistency is very important to users, then we might choose to leave the controls in the new interface for features that were only available on the old appliance. The controls would be disabled to prevent use, but would ensure that the new interface looks exactly like the old interface. Another solution would be to leave blank spaces instead of disabled controls, which would be less confusing to the user but also makes the interfaces less visually consistent. In either of these cases, new controls would be added below the previous interface. If it is only important that common functions be in the same locations, then controls for unavailable features might be replaced with controls for features that are only available on the new appliance.

We are also planning to integrate usage information into these algorithms so that we can ensure that we are making our new interfaces consistent with interfaces that the user is familiar with. An important question that we have not addressed is how much must a user interact with an interface before they will benefit from consistency? How recently must a user have interacted with an interface before the benefits of consistency begin to degrade? Some of this information may be suggested by models of human performance [1]. We also plan to conduct user studies to evaluate these issues.

CONCLUSIONS

We are currently extending our PUC system to enable generation of interfaces that are consistent with previous interfaces the user has interacted with. We are also addressing the multi-device consistency problem. We believe that these two problems share many of the same features and that solving one will suggest solutions for the other.

ACKNOWLEDGMENTS

This work was funded in part by grants from NSF, Microsoft, General Motors, and the Pittsburgh Digital Greenhouse, and equipment grants from Mitsubishi Electric Research Laboratories, VividLogic, Lutron, and Lantronix. The National Science Foundation has funded this work through a Graduate Research Fellowship and under Grant No. IIS-0117658.

REFERENCES

1. Kieras, D. “GOMS modeling of user interfaces using NGOMSL,” in *Conference on Human Factors in Computing Systems.* 1994. Boston, MA: pp. 371-372.
2. Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M. “Generating Remote Control Interfaces for Complex Appliances,” in *UIST 2002.* Paris, France: pp. 161-170.
3. Nichols, J., Myers, B.A., Litwack, K. “Improving Automatic Interface Generation with Smart Templates,” in *Intelligent User Interfaces.* 2004. Funchal, Portugal: pp. 286-288.
4. Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction, Second Edition.* 1992, Reading, MA: Addison-Wesley Publishing Company.