Usable AI: Experience and Reflections

Krzysztof Z. Gajos
Computer Science & Engineering
University of Washington
Seattle, WA 98195
kgajos@cs.washington.edu

Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA 98195
weld@cs.washington.edu

CHI 2008, April 5 – April 10, 2008, Florence, Italy
Workshops and Courses: Usable Artificial Intelligence.

Introduction

We believe that AI has much to offer HCI, in particular allowing for the quick construction of personalized and personalizable interfaces. In this position paper, we report on our experience from four recent investigations of automatic personalization. We then step back and comment on the overall enterprise of making AI usable.

Adaptive Interfaces: Do People Want Them?

Automatic adaptation of user interfaces is a contentious area. Proponents (e.g., [1]) argue that it offers the potential to optimize interactions for a user's tasks and style while critics (e.g., [4]) maintain that the inherent unpredictability of adaptive interfaces may disorient the user, causing more harm than good. Surprisingly, however, there is very little past research explicitly studying automatic adaptation in graphical user interfaces. The existing research includes both positive and negative examples of adaptation, sometimes reporting contradictory results without analyzing the reasons underlying the discrepancy (e.g., [4] and [13]).

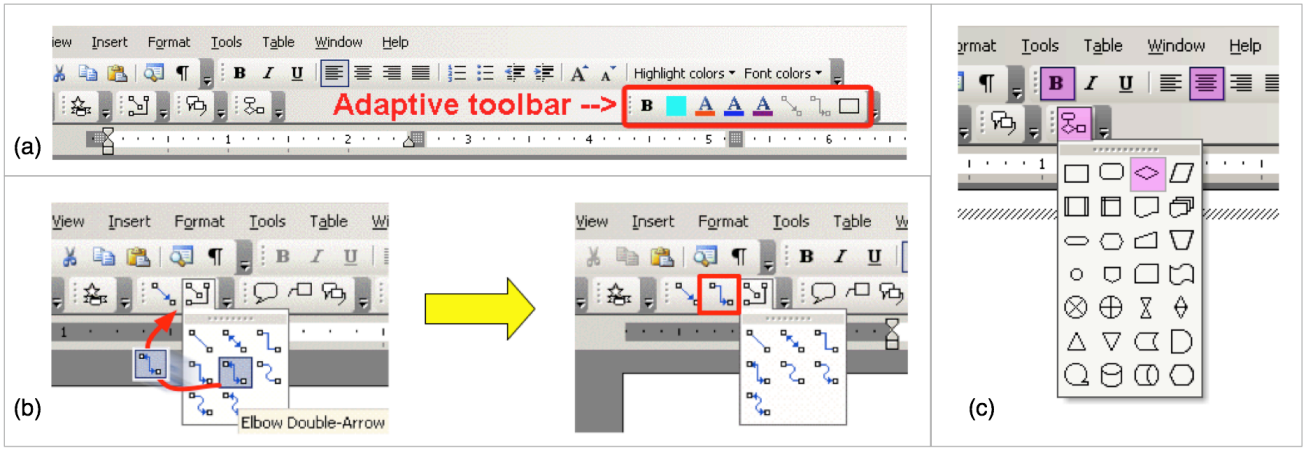


figure 1. Three adaptive interfaces tested in our experiments (as implemented for Microsoft Word): (a) The Split Interface copies frequently used functionality onto a designated adaptive toolbar; (b) The Moving Interface moves frequently used functionality from inside a popup menu to a top level toolbar; (c) The Visual Popout Interface makes frequently used functionality more visually salient.

We have conducted four laboratory studies [5, 7, 8] with two distinct applications (a software graphing calculator and MS Word) and three very distinct adaptation techniques (Figure 1). We have synthesized our results with past research and began to outline how different design choices and interactions make some adaptive interfaces a pleasure to work with while others are frustrating impediments.

In the three studies that directly compared different adaptive techniques, Split Interfaces (where frequently used functionality is copied to a specially designated adaptive part of the interface) were shown to result in significant improvement in both performance and satisfaction compared to the non-adaptive baseline. Our experiments and the analysis of past results also indicated that a number of specific design and context factors impact adoption of adaptive GUIs. Those factors included the accuracy and predictability of the adaptive algorithm, adaptation frequency, the frequency with which the user interacts with the interface, task complexity and the spatial stability of the interface (i.e., to what extent the original interface gets modified during the adaptation).

Of particular interest here is our most recent study [8] where we explored the relative effects of predictability and accuracy in the usability of adaptive interfaces. We say that an adaptive algorithm is predictable if it follows a strategy users can easily model in their heads (we used a random and most recently used strategies to simulate the two ends of the spectrum). We use the term accuracy to refer to the percentage of time that the necessary UI elements are contained in the adaptive area of a Split Interface (we used 50% and 70% accuracy levels). We found that in that particular

design, increasing the adaptive algorithm's accuracy had more beneficial effects on the participants' satisfaction, performance and utilization of the adaptive interface than did improved predictability. These results suggest that there is an opportunity for machine learning approaches to improve both performance and satisfaction of people using adaptive user interfaces, provided those approaches can deliver significant performance improvements over simpler alternatives. The results also pose an AI challenge: algorithms whose behavior appears more predictable (while maintaining high level of predictive accuracy) will result in larger benefit than more opaque approaches.

Automatically Generating Custom Interfaces for Users with Motor Impairments

Users with motor impairments often find it difficult or impossible to use today's common software applications. While many believe that the needs of these users are adequately addressed by specialized assistive technologies, these technologies, while often helpful, have two major shortcomings. First, they are often abandoned, because of their cost, complexity, limited availability and need for ongoing maintenance (it is estimated that less than 60% of the users who need assistive technologies actually use them [3]). Second, assistive technologies are designed on the assumption that the user interface, which was designed for the "average user," is immutable, and thus users with motor impairments must adapt themselves to these interfaces.

We developed an alternative approach: our Supple++ system [10] automatically generates interfaces which are tailored to an individual's motor capabilities and can be easily adjusted to accommodate varying vision

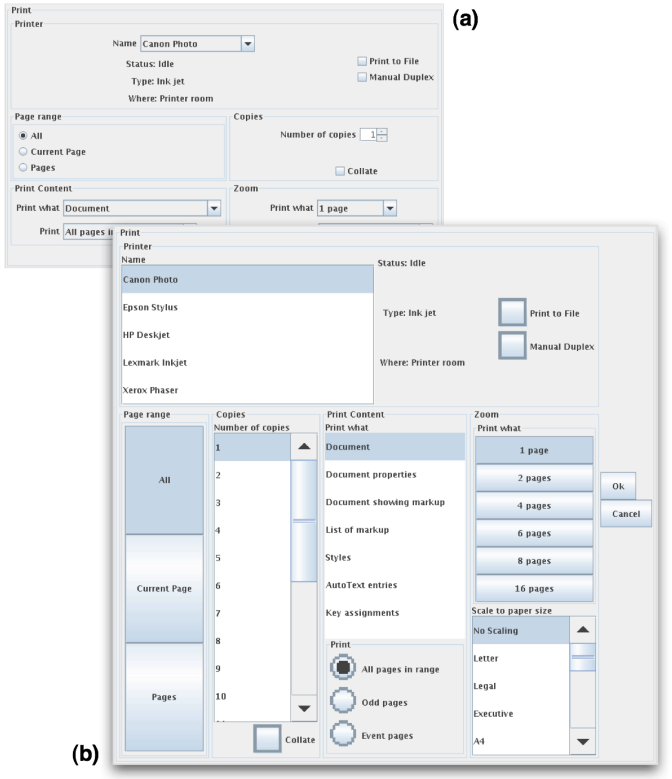


figure 2. (a) The default interface for a print dialog. (b) A user interface for the print dialog automatically generated for a user with impaired dexterity based on a model of her actual motor capabilities.

capabilities (Figure 2). Supple++ uses automatically generated custom regression models to predict users' motor capabilities based on a one-time motor performance test and uses these models in an optimization process, generating personalized interfaces.

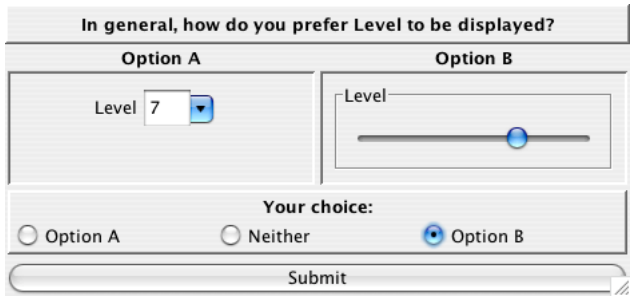
In a study involving 11 participants with motor impairments and 6 able-bodied participants, which compared the automatically generated interfaces to the baselines, our results show that users with motor impairments were 26.4% faster using interfaces generated by Supple++, they made 73% fewer errors, strongly preferred those interfaces to the manufacturers' defaults, and found them more efficient, easier to use, and much less physically tiring [9]. These findings indicate that rather than requiring some users with motor impairments to adapt themselves to software using separate assistive technologies, software can now adapt itself to the capabilities of its users thanks to a combination of AI and HCI innovations.

Preference Elicitation for Interface Optimization
Decision-theoretic optimization is becoming a popular tool in the user interface community, but creating accurate cost (or utility) functions has become a bottleneck — in most cases the numerous parameters of these functions are chosen manually, which is a tedious and error-prone process. Supple's cost functions, for example, typically rely on more than 40 parameters reflecting complex and interacting decision trade-offs. These parameters have to be chosen anew for each new target device and interaction style.

We have thus built Arnauld, a system that allows users to quickly come up with the right parameters just by providing feedback about concrete outcomes [6]. Arnauld uses two types of interactions: system-driven elicitation and user-driven example critiquing.

Users can freely switch between the two types of interactions. During the system-driven elicitation,

(a)



(b)

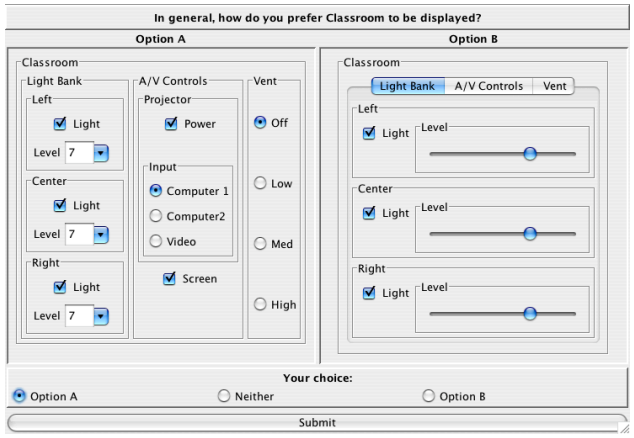


figure 3. Two consecutive steps in the active elicitation process. (a) Arnauld poses a ceteris paribus query, showing two renderings of light intensity control in isolation; this user prefers to use a slider. (b) Realizing that the choice may impact other parts of the classroom controller interface, Arnauld asks the user to consider a concrete interface that uses combo boxes for light intensities but is able to show all elements at once, and an interface where sliders are used but different parts of the interface have to be put in separate tab panes in order to meet the overall size constraints.

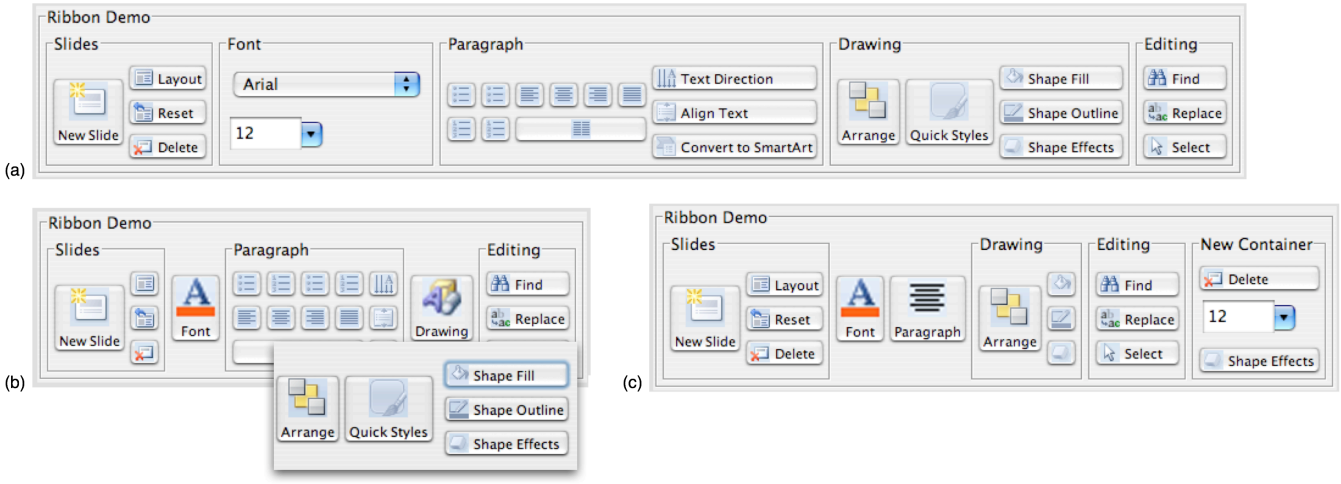


figure 4. MS Ribbon (a) a fragment of the MS Ribbon re-implemented in Supple; (b) Supple automatically provides the size adaptations, which are manually designed in the original version of the MS Ribbon; (c) unlike the manually designed Ribbon, the Supple version allows users to add, delete, copy and move functionality; in this example, New Container section was added, its contents copied via drag-and-drop operations from other parts of the interface and the Quick Style button was removed from the Drawing panel; the customized Supple version of the Ribbon can still adapt to different size constraints.

Arnauld presents the user with a pair of outcomes, always starting with a pair where only one easily identifiable difference exists between the two alternatives (Figure 3a). The user is asked to express preference for one outcome or the other. If the difference causes rippling effects in the larger context, a follow-up query is issued illustrating those effects (Figure 3b). These isolated and situated queries allow Arnauld to identify not only absolute preferences but also trade-offs: for example, in the two queries shown in Figure 3, the user indicated that he preferred sliders to combo boxes but not if they caused the interface to grow so large that it had to be split into separate tab panes.

The example critiquing interaction allows users to change the widget choice or layout of any part of the interface through direct manipulation (using the customization framework, in the case of Supple) – these interactions also provide input to the learning algorithm.

The learning algorithm uses the max-margin approach to find a set of parameters that optimally matches the preferences expressed by the user through the two types of interactions. Related problems have been addressed using Support Vector Machines (requiring solving quadratic optimization problems) [11] and sampling-based algorithms [2]. Because of the interactivity requirements, we developed a novel very

fast algorithm, which only requires solving of a linear optimization problem.

AI Can Improve Usability: The Case of MS Ribbon
Microsoft Ribbon is an interface innovation introduced in MS Office 2007 as a replacement for menus and toolbars. One of it's important properties is that the presentation of the contents of the Ribbon can be adapted based on the width of the document window. The adaptation is performed in several ways, including removing text labels from buttons, re-laying out some of the elements and replacing sections of the Ribbon with pop-up windows. Figure 4a shows a fragment of the Ribbon re-implemented in our Supple system, while Figure 4b shows that same fragment adapted to fit in a narrower window.

The size adaptation of MS Ribbon is not automatic – versions for different window widths were designed by hand. An unfortunate consequence of this approach is that no manual customization of the Ribbon is possible: unlike in the case of toolbars from earlier versions of MS Office, in Ribbon there is no mechanism to allow moving, copying, adding or deleting buttons, panels or other functionality.

We were able to quickly re-implement the Ribbon in Supple. Supple's automatic interface generation algorithm, which takes size as one of the input constraints, automatically provided the size adaptations (Figure 4b). More importantly, however, Supple's built-in customization mechanisms allow people to add new panels to the Supple version of the Ribbon as well as move, copy and delete functionality. The customized Ribbon can be naturally adapted to different size constraints by Supple (Figure 4c). In this case, automatically generated and adapted interactions can improve user's sense of control compared to the manually created solution.

Discussion Points

Stepping back a bit, we now offer some reflections on the enterprise of "making AI usable."

Using AI
Incorporating AI solutions in human-computer interaction brings about many new usability challenges. For example, increased complexity of an AI system's behavior makes it harder for people to create adequate mental models of those systems. As the results our studies of adaptive interfaces suggest, there is a complex trade off between the benefit of improved efficiency of interaction and the increased cognitive complexity of the interactions. In some cases, when the benefits of automation are sufficiently large, those benefits may outweigh the costs and our results demonstrate the need for further research in this area.

Further, noisy input combined with imperfect learning and inference may result in erroneous behavior. Horvitz [12] proposed a decision-theoretic framework for systems to automatically reason about the costs and benefits of potential actions based on the strength of the evidence motivating system's action and an estimate of the actual impact the action would have on the user.

Intelligence Guided by Usage
Similarly, HCI is a source of important, interesting and new challenges for AI. In particular, people...

- are a source of noisy input,

- have strong expectations regarding the predictability of the system's behavior and the correctness of any "intelligent" behavior,
- may want reliable estimates of a system's confidence (if it cannot guarantee uniformly high accuracy),
- often wish for an explanation underlying a recommendation or proposed action, and
- are intolerant of long response times.
- While AI researchers have delivered technologies that meet some of these requirements, they have made little progress on others; thus developing useful and usable systems, will require developing new AI approaches and algorithms.

It is likely to be insufficient to try to add AI to an existing interface, or to try to design a new interface for an existing AI solution. Paraphrasing Dan Olsen: "Choosing a machine learning algorithm independent of the way it will be used by people is an approach doomed to failure." In our own work, we found that we were particularly successful when we considered both aspects of the problem together. This allowed us to create useful and usable systems like Supple++, identify the opportunities for machine learning for adaptive user interfaces, and it inspired us to develop new AI solutions, like the very fast max-margin algorithm for Arnauld or a new optimization-based algorithm for Supple++.

Usable HCI?

Many AI and machine learning techniques have been packaged into toolkits like Weka, Mallet, the Graphical Models Toolkit (GMTK), Crayons, EyePatch, and others, allowing non-AI experts to easily apply those techniques when solving problems in other disciplines. While many problems still require substantial AI expertise, there are many situations where the "typical" application of an existing technique is sufficient; toolkits support those typical applications. The fact that many techniques originating from AI research are now easy to use has spurred important innovations in other fields, including HCI.

It can be argued, however, that even though there are many situations where "typical" HCI methodologies are applicable, those methods haven't been packaged or made accessible to non-HCI experts. For example, even analysis of data collected from a simple user study is fraught with pitfalls: timing data usually has to be log-transformed for analysis with a t-test or ANOVA because these tests are only valid if data is normally distributed; subjective responses should be analyzed with a different set of (non-parametric) tests; if testing for multiple hypothesis at once (a very common situation) p-values need to be corrected to avoid accidentally significant results; if there are missing data, model-based tests may be more appropriate, and so on. Yet, despite the fact that many people conduct similar types of experiments where timing, error and preference data are collected, there are no common and easy to use tools that would help design and analyze such experiments.

We suggest that increasing the usability of the basic HCI methodologies would give non-experts the necessary tools to reason about and effectively communicate the usability properties of inventions that include an interactive component — thus encouraging cross-pollination of the two fields.

References

[1] Benyon, D. Adaptive systems: A solution to usability problems. User Modeling and User-Adapted Interaction, 3, 1 (1993), 65–87.

[2] Chajewska, U., Koller, D., and Ormoneit, D. Learning an agent's utility function by observing behavior. In Proc. of ICML'01. 2001.

[3] Fichten, C., Barile, M., Asuncion, J., and Fossey, M. What government, agencies, and organizations can do to improve access to computers for postsecondary students with disabilities: recommendations based on Canadian empirical data. Int J Rehabil Res, 23, 3 (2000), 191–9.

[4] Findlater, L. and McGrenere, J. A comparison of static, adaptive, and adaptable menus. In Proceedings of ACM CHI 2004. 2004, 89–96.

[5] Gajos, K., Christianson, D., Hoffmann, R., Shaked, T., Henning, K., Long, J. J., and Weld, D. S. Fast and robust interface generation for ubiquitous applications. In Proc. of Ubicomp'05. Tokyo, Japan, 2005.

[6] Gajos, K. and Weld, D. S. Preference elicitation for interface optimization. In Proc. UIST 2005. Seattle, WA, USA, 2005.

[7] Gajos, K. Z., Czerwinski, M., Tan, D. S., and Weld, D. S. Exploring the design space for adaptive graphical user interfaces. In Proc. AVI '06. ACM Press, New York, NY, USA, 2006, 201–208.

[8] Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., and Weld, D. S. Predictability and accuracy in adaptive user interfaces. In submission.

[9] Gajos, K. Z., Wobbrock, J. O., and Weld, D. S. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In submission.

[10] Gajos, K. Z., Wobbrock, J. O., and Weld, D. S. Automatically generating user interfaces adapted to users' motor and vision capabilities. In Proc. UIST '07. ACM Press, 2007, 231–240.

[11] Gervasio, M. T., Moffitt, M. D., Pollack, M. E., Taylor, J. M., and Uribe, T. E. Active preference learning for personalized calendar scheduling assistance. In Proc. IUI '05. ACM Press, 2005, 90–97.

[12] Horvitz, E. Principles of mixed-initiative user interfaces. In Proc. CHI '99. ACM Press, New York, NY, USA, 1999, 159–166.

[13] Sears, A. and Shneiderman, B. Split menus: effectively using selection frequency to organize menus. ACM Trans. Comput.-Hum. Interact., 1, 1 (1994), 27–51.