Decision-Theoretic User Interface Generation

Krzysztof Z. Gajos and **Daniel S. Weld**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195, USA
{kgajos,weld}@cs.washington.edu

Jacob O. Wobbrock
The Information School
University of Washington
Seattle, WA 98195, USA
wobbrock@u.washington.edu

Introduction

For decades, researchers have debated the pros and cons of adaptive user interfaces with enthusiastic AI practitioners often confronting skeptical HCI experts (Shneiderman & Maes, 1997). This paper summarizes the SUPPLE project's six years of work analyzing the characteristics of successful adaptive interfaces and developing efficient algorithms for their automatic generation (Gajos & Weld, 2004; Gajos *et al.*, 2005; Gajos & Weld, 2005; Gajos *et al.*, 2006; Gajos, Wobbrock, & Weld, 2007; Gajos *et al.*, 2008; Gajos, Wobbrock, & Weld, 2008).

We conclude that *personalized user interfaces*, which are adapted to a person's devices, tasks, preferences and abilities, can improve user satisfaction and performance. Further, we demonstrate that automatic generation of these interfaces is computationally feasible.

We developed two systems which autonomously construct a broad range of personalized adaptive interfaces: 1) SUPPLE/ARNAULD uses decision-theoretic optimization to automatically generate user interfaces adapted to target device, usage patterns, and a learned model of user preferences, and 2) SUPPLE++ first performs a one-time assessment of a person's motor abilities and then automatically generates user interfaces adapted to that user's abilities.

Our experiments show that these automatically generated interfaces significantly improve speed, accuracy and satisfaction of users with motor impairments compared to manufacturers' defaults.

We also provide the first characterization of the design space of adaptive interfaces and demonstrate how such interfaces can significantly improve the quality and efficiency of daily interactions for typical users.

User Interface Generation as Optimization

Our SUPPLE system (Gajos & Weld, 2004) uses constrained decision-theoretic optimization to automatically generate user interfaces. As input, SUPPLE takes a *functional specification* of the interface, which describes the types of information that need to be communicated between the application and the user, the device-specific *constraints*, such as screen size and a list of available interactors, a typical *usage trace*, and a *cost function*. The cost function can correspond to any measure of interface quality, but we have focussed on conformance to user preferences and expected speed of operation (see below). SUPPLE's optimization algorithm, a branch-and-bound search with constraint propagation and a novel admissible heuristic, is guaranteed to find the user interface which minimizes the cost function while also satisfying all device constraints.

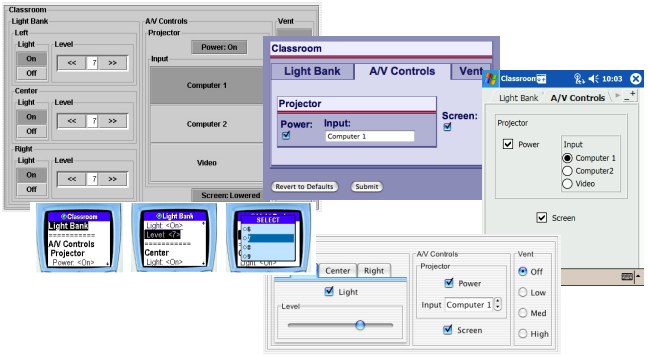

Figure 1: An interface for a simple application rendered automatically by SUPPLE for a touch panel, an HTML browser, a PDA, a desktop computer and a WAP phone.

Despite the huge space of possible designs (exponential in the number of interface elements), our algorithm renders even complex interfaces in less than two seconds on a standard desktop computer, thus enabling fast and personalized delivery of user interfaces.

Although there is some previous work that used optimization methods for laying out widgets within a dialog box (e.g., Fogarty & Hudson, 2003; Sears, 1993), our rendering algorithm does much more: it chooses the widgets, it chooses the navigation structure of the UI (i.e., puts things into tab panes or pop-up windows if not everything can fit on one screen), and chooses the layout. Furthermore, SUPPLE's optimization approach is more robust than previous rule-based techniques for UI generation (e.g., Nichols *et al.*, 2002). For example, SUPPLE naturally adapts to devices with vastly different screen sizes (Gajos & Weld, 2004; Gajos *et al.*, 2005). Furthermore, in order for SUPPLE to render interfaces for a new kind of device, one need only specify the widgets available on the device and a cost function (Figure 1). The next two sections describe how to automatically learn these cost functions.

Adapting to User Preferences

By providing a different cost (utility) function, one can direct SUPPLE to produce very different styles of user interfaces. For example, by encoding a user's preferences in the cost function, SUPPLE would generate user interfaces that the user favored. However, SUPPLE's cost functions typically rely on more than 40 parameters reflecting complex decision trade-offs. Manual selection of the values of these parameters is a tedious and error-prone process. To address this we built ARNAULD, a system that quickly calculates parameter values from user feedback about concrete outcomes (Gajos & Weld, 2005). To do this ARNAULD uses two types of interactions: *system-driven elicitation* with pairwise comparison queries (Figure 2) and *user-driven example critiquing*, which relies on user-initiated improvements to the SUPPLE-generated interfaces as input for the learning system. These interactions allow people to express their preferences more easily and consistently than other common approaches such as ranking or rating (Carterette *et al.*, 2008) but they also necessitated development of a novel machine learning algorithm.

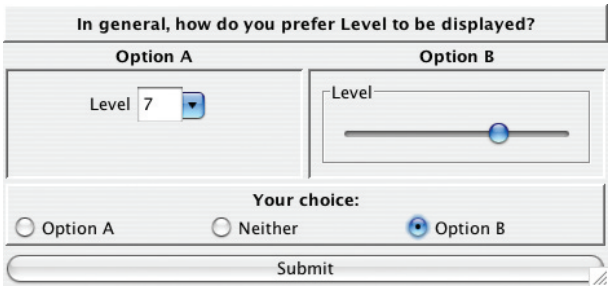

Figure 2: The first step in the active elicitation process: ARNAULD poses a *ceteris paribus* query, showing two renderings of light intensity control in isolation; this user prefers to use a slider. Realizing that the choice may impact other parts of the classroom controller interface, ARNAULD will subsequently ask 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.

ARNAULD's learning algorithm uses a max-margin approach to find a set of parameters that optimally matches the preferences expressed by the user. Similar problems have been addressed using Support Vector Machines (Gervasio *et al.*, 2005), but this requires expensive quadratic optimization. Bayesian reasoning using Manhattan sampling has also been tried (Chajewska, Koller, & Ormoneit, 2001). Because none of these approaches were fast enough for interactive use *we developed a faster algorithm, recasting the problem as linear optimization*.

The system-driven, elicitation queries require ARNAULD to generate the most informative comparisons for the user's consideration — i.e., it is an active learning problem. We developed two heuristic, query-generation algorithms: one general, which performs computation entirely in the parameter space, and the other a domain-specific approach, which is more robust to inconsistencies in user responses. Our user and algorithmic evaluations show that only 40-55 user responses are needed to learn a cost function that closely approximates the desired behavior.

Adapting to Motor and Vision Capabilities

An alternative to ARNAULD's model of a users' qualitative preferences is a model of their quantitative *motor performance*, that is the speed with which they can accomplish a set of tasks. Our SUPPLE++ system (Gajos, Wobbrock, & Weld, 2007) automatically generates interfaces tailored to an individual's actual motor abilities (Figure 3 right). Specifically, SUPPLE++'s Activity Modeler administers a one-time performance test, then performs automatic feature selection to generate an individual regression model of user's motor abilities. SUPPLE++ then uses the model to create UIs that minimize the expected time it will take the user to complete a set of typical tasks.

Allowing variable-sized widgets was crucial to accommodate many user's pointing profiles, but this additional variable greatly expands the size of the search space explored during rendering. Furthermore, in our original formulation, the effectiveness of our admissible heuristic guiding the search relied on a cost function that could be factored in such a way that the cost of each widget or layout decision could be computed independently of others. However, in order to compute the expected *time* needed to navigate through a particular interface layout, movement distance and the size of the target need to be known, making it impossible to compute the cost of a layout without knowing all the interface elements comprising that layout. Ensuring quick interface-generation times with a performance-based cost function required significant modifications to the optimization algorithm and a novel admissible heuristic (Gajos, Wobbrock, & Weld, 2007).

SUPPLE++ also allows users to adjust the interfaces to their vision capabilities, and the two adaptations — to motor and vision capabilities — can be used in combination. This is an important contribution because users with low vision *and* low dexterity are poorly supported by existing assistive technologies.

Besides producing adaptations for people with motor impairments, SUPPLE++ provides a simple approach for adapting interfaces to unusual devices or interaction techniques, such as touch panels or projected displays where a laser pointer controls the mouse cursor.

Impact for Users with Motor Impairments

This technology has the potential to provide great benefit. In a study involving 11 participants with motor impairments, which compared the interfaces generated by SUPPLE++ to the default designs, the results show that users with motor impairments were 26% faster using interfaces generated by SUPPLE++, 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 (Gajos, Wobbrock, & Weld, 2008).

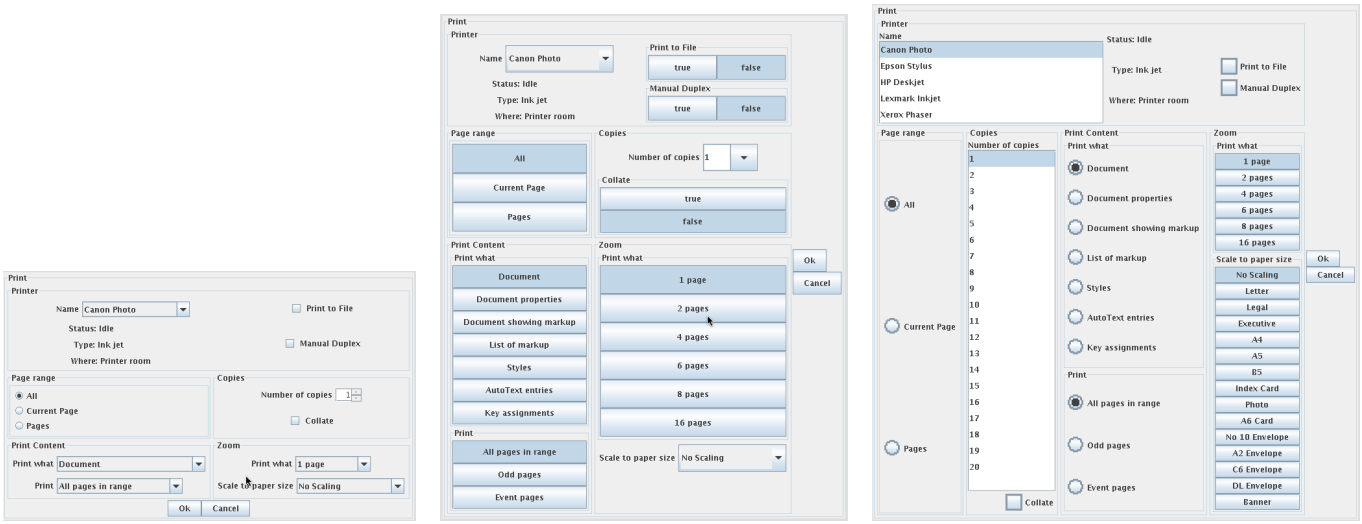

Figure 3: Three user interfaces for a print dialog box: a default reproduced from Microsoft Word 2003 and two automatically generated for a user with impaired dexterity: one based on his subjective preferences as elicited by ARNAULD, and one based on his actual motor abilities as modeled by SUPPLE++.

While some may contend 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 (Dawe, 2006; Fichten *et al.*, 2000; Phillips & Zhao, 1993) and it is estimated that only about 60% of the users who need assistive technologies actually use them (Fichten *et al.*, 2000). Indeed, while we gave our participants the option to use any input devices they wanted, all of them chose either a mouse or a trackball. 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 by using specialized devices (Keates *et al.*, 2002)

In contrast, our results demonstrate that users with motor impairments can perform well with conventional input devices (such as mice or trackballs) if given interfaces that accommodate their unique motor abilities. These results also show that automatic generation of user interfaces based on users' motor abilities is feasible, and that the resulting interfaces are an attractive alternative to manufacturer's defaults.

We were particularly struck by the situation of one participant, who controls the trackball with his chin and types on a keyboard with a head-mounted wand, making keyboard shortcuts inconvenient. Because his speech is also impaired, he cannot use speech recognition software. He works as an IT consultant so his livelihood is critically dependent on being able to interact with computers effectively. Currently, he has to compensate with perseverance and long hours for the mismatch between the current state of technology and his abilities. He was the slowest participant in our study when using the manufacturers' default interfaces, but with SUPPLE++ he was able to close nearly half of the performance gap between himself and our able-bodied participants.

Adapting to Common Tasks

In previous sections we discussed strategies for long term interface adaptations, where once created, the interfaces are not expected to change unless such a change is explicitly requested by the user. A complementary approach is to consider interfaces, which adapt continuously to user's immediate task at hand. Such adaptation of user interfaces is a contentious area. Proponents (e.g., Benyon, 1993) argue that it offers the potential to optimize interactions for a user's tasks and style, while critics (e.g., Findlater & McGrenere, 2004) maintain that the inherent unpredictability of adaptive interfaces may disorient the user, causing more harm than good. Surprisingly, however, little empirical evidence exists to inform this debate, and the existing research includes both positive and negative examples of adaptation, sometimes reporting contradictory results without analyzing the reasons underlying the discrepancy (e.g., Findlater & McGrenere (2004) and Sears & Shneiderman (1994)).

We have conducted four laboratory studies with two distinct applications and three very distinct techniques for automatically adapting an interface to the person's current task at hand.

By synthesizing our results with past research, *we provided the first characterization of the design space of adaptive graphical user interfaces*, showing how different design and contextual factors affect users' performance and satisfaction. Specifically, we studied 1) several adaptation interaction models, 2) accuracy and predictability of adaption, 3)

the adaptation frequency, 4) the frequency with which the user interacts with the interface, and 5) the task complexity. See (Gajos *et al.*, 2005, 2006, 2008) for complete results.

Our studies consistently showed that one approach to adaptation, which we call *Split Interfaces* (Figure 4), results in a *significant improvement in both performance and satisfaction* compared to the non-adaptive baselines. In Split Interfaces, frequently-used functionality is automatically copied to a specially designated adaptive part of the interface. This allows the user to either follow their familiar route or potentially save time by exploiting the adaptation.

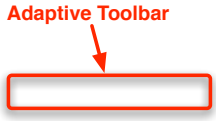

Figure 4: In a *Split Interface* functionality that is predicted to be immediately useful is copied to a designated adaptive part of the interface.

Of particular interest to the AI community may be our results concerning the trade-off between predictability and accuracy of the adaptive algorithm (Gajos *et al.*, 2008). We say that an adaptive algorithm is *predictable* if it follows a strategy users can easily model in their heads (such as promoting the most recently used items). We use the term *accuracy* to refer to the percentage of time that the necessary UI elements are contained in the adaptive area. We found that in our 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. This suggests that the benefits of a large improvement in accuracy may outweigh the disadvantages of decreased predictability. However, the analysis of our eye-tracking data indicates that more predictable adaptive algorithms help users better anticipate when helpful adaptations *might* have taken place, which likely helps reduce the visual-search time as well as the cognitive effort involved in using the adaptive interface.

Adaptation to Tasks in SUPPLE

Informed by our studies, we implemented the Split Interface approach in SUPPLE for adapting to the user's task at hand. Unlike previous implementations of this general approach, which could only adapt contents of menu items or toolbar buttons, SUPPLE can adapt *arbitrary* functionality: frequently-used but hard to access functionality is copied to the functional specification of the adaptive area and SUPPLE automatically renders it in a manner that is appropriate given the amount of space available in the adaptive part of the interface (Gajos *et al.*, 2005). For example, if the user frequently changes the print quality setting (which requires 4 to 6 mouse clicks to access in a typical print dialog box), SUPPLE will copy that functionality to the adaptive part of the main print dialog box (Figure 5, left).

Our decision-theoretic approach also allows SUPPLE to *adapt to a person's long term usage patterns*. By reweighing the terms of the cost function in keeping with collected usage traces, SUPPLE generates interfaces where frequently used functionality gets rendered using more convenient widgets, and where elements that are frequently used together are rendered side by side rather than in separate windows or panes (Gajos & Weld, 2004).

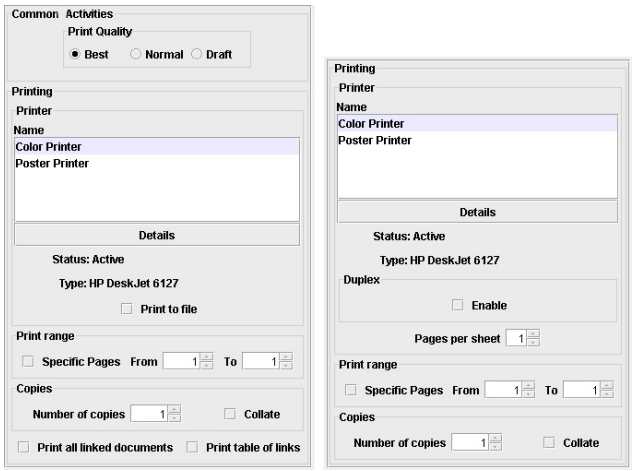

Figure 5: Two examples of personalization in SUPPLE: the left window features a dynamic section at the top whose automatically updated content reflects the most common activity; the right window was customized by the user: she removed some elements (e.g., "Print to file") and added "Duplex" and "Number of pages per sheet" elements by dragging them from a separate Printer Properties window.

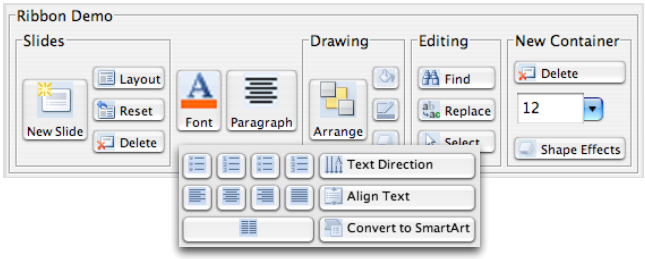

Figure 6: Unlike Microsoft's manually designed Ribbon, the SUPPLE version allows users to add, delete, copy and move functionality; in this example, the "New Container" section was added, its contents copied via drag-and-drop operations from other parts of the interface, and the "Quick Style" button was removed from the "Drawing" panel; the customized SUPPLE version of the Ribbon can still adapt to different size constraints.

Finally, SUPPLE supports extensive *user-directed customization*. With a mouse right-click, users can change the presentation of a particular piece of functionality (e.g., request that a slider, rather than a pull-down menu, be used to represent sound volume), layout or navigation structure (e.g., request that particular set of items be rendered side-by-side rather than in separate tab panes or pop-up windows). These customizations are then represented as additional constraints during the interface optimization process. Users can also copy, move or delete parts of the interface. For example, instead of relying on automatic adaptation, a user could copy the duplex control for a printer from the "Properties"

pop-up window to any place in the main print dialog (Figure 5, right). Those customizations are recorded as modifications to the functional specification and thus can be applied even when the interface is subsequently rendered on a different device.

This level of user control over presentation is not currently possible in manually designed user interfaces. A case in point is the *Ribbon*, an interface innovation introduced in Microsoft Office 2007. Ribbon, which replaces toolbars and menus, adapts to the width of the window, but the adaptation is not automatic — many versions of the Ribbon were manually designed for different ranges of window sizes. An unfortunate consequence of this approach is the *Ribbon's inability to support user customization!* Unlike toolbars from earlier versions of Microsoft Office, Ribbon has no mechanism for end-users to move, copy, add or delete buttons, panels or other functionality. Our SUPPLE implementation of Ribbon (Gajos & Weld, 2008) naturally supports both the adaptation to window width and a rich set of customizations, as illustrated in Figure 6.

Conclusion

The work summarized in this paper brings new evidence to the long-running debate over adaptive user interfaces and has important ramifications for AI researchers. The algorithms presented in Gajos & Weld (2004) and Gajos, Wobbrock, & Weld (2007) show that decision-theoretic optimization can quickly generate personalized interfaces. The experiments of Gajos, Wobbrock, & Weld (2008) demonstrate that certain classes of users (e.g. those with motor impairments) strongly prefer and perform faster with automatically generated interfaces. The framework described in Gajos *et al.* (2006) provides strong guidance for those designing adaptive interfaces, since some strategies are appreciated by users while other types can cause major frustration. Many in the HCI community have warned that machine learning can be dangerous in interfaces, because the unpredictably-complex behavior may confuse users, but the study presented in Gajos *et al.* (2008) shows that users may be willing to sacrifice predictability if the alternative has a large-enough improvement in adaptation accuracy. Automatic interface generation and adaptation promises many advantages, and we urge more researchers to investigate these important issues.

References

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

Carterette, B.; Bennett, P. N.; Chickering, D. M.; and Dumais, S. T. 2008. Here or there: Preference judgments for relevance. In *Proceedings of the European Conference on Information Retrieval (ECIR)*. To appear.

Chajewska, U.; Koller, D.; and Ormoneit, D. 2001. Learning an agent's utility function by observing behavior. In *Proceedings of ICML'01*.

Dawe, M. 2006. Desperately seeking simplicity: how young adults with cognitive disabilities and their families adopt assistive technologies. *Proceedings of the SIGCHI conference on Human Factors in computing systems* 1143–1152.

Fichten, C.; Barile, M.; Asuncion, J.; and Fossey, M. 2000. 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):191–9.

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

Fogarty, J., and Hudson, S. E. 2003. GADGET: A toolkit for optimization-based approaches to interface and display generation. In *Proceedings of UIST'03*.

Gajos, K., and Weld, D. S. 2004. Supple: automatically generating user interfaces. In *Proceedings of the 9th international conference on Intelligent user interface*, 93–100. Funchal, Madeira, Portugal: ACM Press.

Gajos, K., and Weld, D. S. 2005. Preference elicitation for interface optimization. In *Proceedings of UIST 2005*.

Gajos, K. Z., and Weld, D. S. 2008. Usable AI: Experience and reflections. In *Workshop on Usable Artificial Intelligence (at CHI'08)*.

Gajos, K.; Christianson, D.; Hoffmann, R.; Shaked, T.; Henning, K.; Long, J. J.; and Weld, D. S. 2005. Fast and robust interface generation for ubiquitous applications. In *Proceedings of Ubicomp'05*.

Gajos, K. Z.; Czerwinski, M.; Tan, D. S.; and Weld, D. S. 2006. Exploring the design space for adaptive graphical user interfaces. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, 201–208. New York, NY, USA: ACM Press.

Gajos, K. Z.; Everitt, K.; Tan, D. S.; Czerwinski, M.; and Weld, D. S. 2008. Predictability and accuracy in adaptive user interfaces. In *CHI'08*. New York, NY, USA: ACM Press.

Gajos, K. Z.; Wobbrock, J. O.; and Weld, D. S. 2007. Automatically generating user interfaces adapted to users' motor and vision capabilities. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, 231–240. New York, NY, USA: ACM Press.

Gajos, K. Z.; Wobbrock, J. O.; and Weld, D. S. 2008. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In *CHI'08*. New York, NY, USA: ACM Press.

Gervasio, M. T.; Moffitt, M. D.; Pollack, M. E.; Taylor, J. M.; and Uribe, T. E. 2005. Active preference learning for personalized calendar scheduling assistance. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, 90–97. New York, NY, USA: ACM Press.

Keates, S.; Langdon, P.; Clarkson, J. P.; and Robinson, P. 2002. User models and user physical capability. *User Modeling and User-Adapted Interaction* 12(2):139–169.

Nichols, J.; Myers, B.; Higgins, M.; Hughes, J.; Harris, T.; Rosenfeld, R.; and Pignol, M. 2002. Generating remote control interfaces for complex appliances. In *Proceedings of UIST'02*.

Phillips, B., and Zhao, H. 1993. Predictors of assistive technology abandonment. *Assist Technol* 5(1):36–45.

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

Sears, A. 1993. Layout appropriateness: A metric for evaluating user interface widget layout. *Software Engineering* 19(7):707–719.

Shneiderman, B., and Maes, P. 1997. Direct manipulation vs. interface agents. *Interactions* 4(6):42–61.