Improving User Interface Personalization

Krzysztof Gajos, Raphael Hoffmann and Daniel S. Weld
University of Washington
Seattle, WA, USA
{kgajos,raphaelh,weld}@cs.washington.edu

ABSTRACT

SUPPLE uses decision-theoretic optimization to render an abstract *functional specification* into an adaptive interface, which is personalized both to an individual's usage pattern and the characteristics of a target device. This paper briefly describes three enhancements to SUPPLE: 1) light-weight utility elicitation, 2) the ability to adapt to user's behavior by generating interfaces with multiple ways to access the same functionality, and 3) generalization-based customization.

Categories and Subject Descriptors: D.2.2 [**Design Tools and Techniques**]: User Interfaces, H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUIs)

INTRODUCTION

Interfaces shipped with today's complex applications are designed in a "one size fits all" manner; alas, by aiming to address the needs of the "average user" they miss *essential* needs of most individual users. In contrast, we believe each user deserves a custom-built UI that best reflects her needs. Realizing this dream is complicated by the shift away from the desktop and toward pervasive computing. Most of today's applications are designed to work with keyboard and pointer and assume a small range of screen sizes. However, people are using an increasing variety of display-equipped devices, which employ different interaction techniques and span a huge range of display sizes (*e.g.*, cell-phones to live boards).

In response, we are creating SUPPLE, a system that generates an interface which optimizes the user's expected utility on the device at hand and adapts as appropriate to changes in user activity. After a brief summary of SUPPLE as presented in [3], we describe our recent progress:

- Since SUPPLE's behavior depends on an accurate estimate of the user's utility function, we use every interaction (*e.g.*, user's customization commands) to refine the system's utility estimate.
- SUPPLE can dynamically create "one touch" access to commonly-used functionality, while maintaining a stable, predictable interface structure.
- SUPPLE's flexible customization facility uses machine learning to interpret a user's request as potentially applying to more than one aspect of an interface, *e.g.*, perhaps to multiple applications.

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA
ACM 1-58113-962-4/04/0010

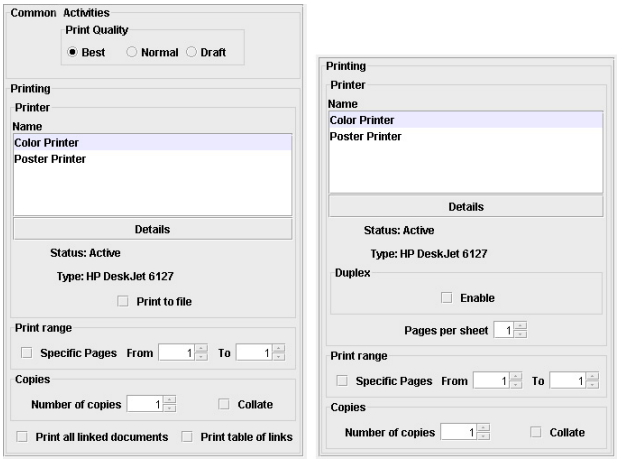

Figure 1: Two examples of personalization in SUPPLE: the left window features a dynamic section at the top whose automatically updated content reflects the most common activity; the right window was customized by the user: some functionality was removed while duplex printing and number of pages per sheet were added.

REVIEW: ADAPTATION AS OPTIMIZATION

We cast the user interface generation and adaptation as a decision-theoretic optimization problem, where the goal is to minimize the estimated user effort for manipulating a candidate rendering of the interface. SUPPLE takes three inputs: a *functional interface specification*, a *device model* and a *user model*. The functional description defines the *types* of data that need to be exchanged between the user and the application. The device model describes the widgets available on the device, as well as cost functions, which estimate the user effort required for manipulating supported widgets with the interaction methods supported by the device. Finally, we model a user's typical activities with a device- and rendering-independent *user trace*. Details of these models and rendering algorithms are available in [3].

UTILITY ELICITATION

User interface design is an inherently subjective activity. The final choice is a matter of personal taste and preference. Thus, instead of specifying the parameters of the cost function exactly, we provide a loosely specified cost function that already produces reasonable results but can be tuned to the needs of a particular user. The challenge is to find a framework that allows SUPPLE to perform the fine-tuning with minimal input from or disturbance to the user. Currently we are using the minimax regret methods [1] for calculating optimal UI from a loosely specified cost function and for find-

ing a small number of binary queries (*i.e.*, "which of the two renderings do you prefer?") to ask of the user to refine the cost function parameters. In our previous work, we used user traces to vary the weights assigned to different components of the cost function [3] while this new method allows us to vary the relative preference for different widgets.

ADAPTING TO USER BEHAVIOR – DYNAMICALLY

SUPPLE strives to provide an optimal rendering of a UI for a user. Consequently, as it learns more about the user's activities, it should adapt the rendered UI to better serve the user. Previously, we demonstrated how to produce different renderings of the same hierarchically-defined UI given differing traces of user actions [3]. While re-rendering the entire UI might provide the user with a better tool, it may also lead to confusion. Indeed, Shneiderman [5] and others have emphasized the importance of the user's perception of control in HCI, and advocated interface *stability* to improve the user's sense of control. Yet an interface which makes common actions cumbersome is frustrating, even if stability enables familiarity. There is a fundamental tension between *stability* and adaptation towards an *optimal* organization for a user's ongoing behavior.

Our principle of partitioned dynamicity manages the stability-adaptivity tradeoff by segmenting an interface into static and dynamic areas [7]. The static area provides stable navigation to all application functionality, while the dynamic part may adapt to user activity, providing convenient "one touch" access to frequently used functionality. For example, the new Start menu in Windows XP illustrates this approach: one can always reach all the programs through the "All Programs" button, but frequently used items are shown in a separate panel, whose contents change with time.

SUPPLE now uses partitioned dynamicity to organize the adaptive interfaces it creates: every time it renders a concrete UI, it creates a rendering that corresponds exactly to the hierarchical organization of the functional specification. In addition, in each window (or a page on a cell phone), it sets aside some space for dynamic content. As it learns more about the user's behavior, SUPPLE adapts the contents of the dynamic sections of each window. The dynamic content may include duplication of functionality which exists elsewhere in the UI (*e.g.*, adding widgets for controlling two-sided printing on the main print dialog window) or navigational shortcuts in the form of buttons or hyperlinks that take the user to a different part of the UI that normally would require several clicks on links, tabs, buttons, menus, etc.

Within this paradigm, there are two main strategies we are exploring: adapting to user's average behavior and adapting to the immediate task at hand. In the first approach, SUPPLE occasionally updates the dynamic part of the UI with elements that are predicted to save the user most effort over extended periods of time (Figure 1). The second strategy dictates that SUPPLE tries to predict user's immediate actions and change the dynamic part of the UI frequently in the anticipation of the next action. So far we have implemented two algorithms for adapting to user's average behavior: a slow but provably optimal algorithm and a much faster greedy approximation. Our initial experiments show that in practice the greedy algorithm also tends to produce optimal solutions.

GENERALIZED CUSTOMIZATION

SUPPLE's base customization capabilities are powerful: users can delete, copy, or move any piece of functionality to any other part of the UI (*e.g.*, as in Figure 1). Any dialog window can be automatically skipped, as long as the user can confidently set default values for required parameters.

In addition, SUPPLE has the novel capability to *generalize* user customizations, proposing additional changes to the user. For example, if a user customizes a print dialog box in one application, similar customizations are proposed in other applications. SUPPLE uses machine learning techniques, in particular version space algebra [4], to quickly induce a *scope* describing a set of similar situations where a given customization might be desirable.

RELATED WORK

Our work builds on earlier research on model-based user interface generation [6], but differs in two important aspects. 1) In contrast to previous rule-based approaches, we use optimization to select widgets, design the navigation structure, and lay out the elements. 2) Our *functional specification* is at an intermediate level of abstraction between previously described "task specifications" and "abstract specifications" enabling SUPPLE to make flexible rendering decisions, while still allowing the designer to specify the UI's behavior. Eisenstein et al. demonstrated the usefulness of refining the utility function in the rule-based approach [2].

CONCLUSIONS AND FUTURE WORK

We are designing user studies to explore different adaptation strategies and to evaluate the benefits of automatically generalizing customizations. In the future, we plan to endow SUPPLE with the capability to explain its decisions to the users. Perhaps the biggest barrier to adoption is SUPPLE's lack of an authoring tool, allowing designers to create interfaces in a way that preserves SUPPLE's flexibility, yet allows the designer to feel in control of the final product.

Acknowledgements This work supported by NSF grant IIS-0307906 and ONR grant N00014-02-1-0932. Thanks to James Landay, Kate Deibel and Anthony Wu for comments.

REFERENCES

1. C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Regret-based utility elicitation in constraint-based decision problems. Working Paper.
2. J. Eisenstein and A. Puerta. Adaptation in automated user-interface design. In *IUI'00*, New Orleans, LA, 2000.
3. K. Gajos and D. S. Weld. Supple: automatically generating user interfaces. In *IUI'04*, Funchal, Portugal, 2004.
4. T. Lau, P. Domingos, and D. S. Weld. Version space algebra and its application to programming by demonstration. In *ICML'00*, June 2000.
5. B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 3rd edition, 1998.
6. P. Szekely. Retrospective and challenges for model-based interface development. In *Design, Specification and Verification of Interactive Systems '96*, Wien, 1996.
7. D. S. Weld, C. Anderson, P. Domingos, O. Etzioni, K. Gajos, T. Lau, and S. Wolfman. Automatically personalizing user interfaces. In *IJCAI03*, Acapulco, Mexico, 2003.