The Need for an Interaction Cost Model in Adaptive Interfaces

Bowen Hui
Dept. of Computer Science
University of Toronto
bowen@cs.utoronto.ca

Sean Gustafson, Pourang Irani
Dept. of Computer Science
University of Manitoba
{umgusta1,irani}@cs.umanitoba.ca

Craig Boutilier
Dept. of Computer Science
University of Toronto
cebly@cs.utoronto.ca

ABSTRACT

The development of intelligent assistants has largely benefited from the adoption of decision-theoretic (DT) approaches that enable an agent to reason and account for the uncertain nature of user behaviour in a complex software domain. At the same time, most intelligent assistants fail to consider the numerous factors relevant from a human-computer interaction perspective. While DT approaches offer a sound foundation for designing intelligent agents, these systems need to be equipped with an *interaction cost model* in order to reason the impact of how (static or adaptive) interaction is perceived by different users. In a DT framework, we formalize four common interaction factors — information processing, savings, visual occlusion, and bloat. We empirically derive models for bloat and occlusion based on the results of two users experiments. These factors are incorporated in a simulated help assistant where decisions are modeled as a Markov decision process. Our simulation results reveal that our model can easily adapt to a wide range of user types with varying preferences.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Miscellaneous; I.2.11 [**Artificial Intelligence**]: Intelligent agents

General Terms

Interaction models and techniques, User interaction studies

Keywords

Information processing, visual occlusion, bloat, perceived savings

1. INTRODUCTION

Software customization has become increasingly important as users are faced with larger, more complex applications. For a variety of reasons, software must be tailored to specific individuals and circumstances [14]. Online and automated help systems are becoming increasingly prevalent to assist users identify and master different software functions [11]. In this paper, we focus on *adaptive interfaces* where the user's preferences over interface attributes (e.g., location, transparency, perceived savings of interface widgets) determine how the system customizes the interface. Our objective is to develop adaptive interfaces that maximizes the user's ease of interaction with the system.

Many decision-theoretic (DT) approaches have been applied to develop assistants that provide intelligent help for different users (e.g., [11, 1, 6, 3, 2, 13]). These approaches typically try to help the user accomplish a task more efficiently by using machine learning techniques to estimate user-specific information, such as the user's current task, whether the user needs help, or how frustrated the user is with the system. At the same time, every system action has a value (i.e., cost or benefit) that may be perceived differently depending on the user or the circumstance. In support of DT approaches, Horvitz proposed that a central principle in designing intelligent systems is the ability to evaluate the *utility* of system actions "in light of costs, benefits, and uncertainties" [10]. Following these approaches, we adopt a DT framework to design an agent that makes rational decisions about its customization under uncertainty.

In order to model the utility of system actions, we need to directly account for the impact that the system's customization actions have on the user. Since an interface is a composition of widgets, we design the system to adapt the interface by changing the attributes of individual widgets. We refer to such changes as *system actions*, which an intelligent agent may decide to take. However, different actions have different consequences: an adaptive interface that hides unused menu items may be preferable for one user because it saves him from scanning unnecessary functions (i.e., savings from processing extraneous items), but the same behaviour may be detrimental to other users who prefer to see all available functions (i.e., high tolerance to bloat). Furthermore, system actions have effects beyond immediate consequences. For example, a user who likes unused menu items hidden may find it annoying when he needs to use one of those functions in the future (i.e., cost of re-discovery). These consequences are, in fact, ways that a user determines the level of satisfaction with a software. Therefore, to quantify the impact of system actions, we identify interaction factors that are relevant to intelligent interfaces and formalize the costs and benefits of adaptive actions using an *interaction cost model*.

The benefits of developing an explicit interaction cost model are two-fold. From an HCI perspective, quantitative models enable designers to compare a variety of interaction mechanisms on the same grounds and predict the performance and satisfaction that new users experience with these mechanisms. From an intelligent systems perspective, the interaction cost model provides a way for the system to evaluate the utility of its own actions *before* adapting the interface. In developing DT systems, we employ the interaction cost model to evaluate the impact of different adaptive actions. By adopting the formalism that takes *user types* as parameters in the interaction cost model [13], the agent is able to quantify the im-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AVI '08 Napoli, Italy
Copyright 2008 ACM 0-12345-67-8/90/01 ...$5.00.

pact of its actions with respect to specific types of users, and thus, adapting its overall behaviour toward specific user types.

In Section 2, we describe the approach of an intelligent system in a DT framework. Our focus is on modeling the utility of system actions. For this purpose, we identify four common interaction factors for adaptive interfaces — information processing, savings, visual occlusion, bloat — and formalize their cost models. In an effort to derive a quantitative model for occlusion and bloat, we conduct two experiments to explore the relevant parameters and structure in Section 3. Using these experiment results, we implement our interaction cost model in a simulated help system that adds or deletes unused menu items. The simulation is implemented as a *Markov decision process* (MDP) [17]. Our results show that the system is able to adapt its behaviour to different types of users. While usability studies are needed to confirm our simulation results, this work suggests that using a DT framework to model interaction benefits and costs is a formal, general, and useful approach.

2. DECISION-THEORETIC FRAMEWORK

In designing an adaptive interface system, we view the system as an intelligent agent that reasons about the impact that its actions have on the user. Considering adaptive menus in the context of interface bloat as an example, the agent observes which menu items have been selected, evaluates the (long term) utility of hiding one or more unused menu items, and carries out the action that is optimal for the user (i.e., (un)hiding menu items or doing nothing). In general, there are uncertainties in assessing the utility of system actions — how much faster is it for the user to search in that menu after hiding unused items, how tolerant is the user with respect to bloated interfaces, or how likely, and at what frequency, is the user going to require a hidden menu item in the (near) future? These questions illustrate the need to quantify two parts of the customization problem: (i) the costs and benefits of actions with respect to relevant interaction factors (e.g., "how fast", "how tolerant"), and (ii) the likelihood of events (e.g., the probability that the user is highly tolerant to bloat, the probability of a hidden menu item being executed in the near future). Since the focus of this work is utility assessment, we will assume probability estimation is feasible in the system.[1] Section 2.1 explains the concept of utility and its relation to user preferences. In Section 2.2, we turn to the discussion of adaptive interface systems and relevant interaction factors. Section 2.3 formalizes our interaction cost model and explains how it is used in the DT framework.

2.1 Objective Value versus Subjective Utility

Utility theory is used to systematically quantify the total costs and benefits of decision outcomes: if a person has higher utility in one situation than another, that person prefers the former situation over the latter. In designing intelligent systems, we want to know the utility of adaptive actions in a way that reflects the user's preferences over possible interaction mechanisms. Intuitively, our goal is to quantify the utility of specific system actions with respect to the factors that influence the user's interaction experience. Note that utility is subjective in nature since it reflects individual preferences. Therefore, our goal is to determine the perceived utility given an interaction setting (i.e., in terms of system actions and application states). To do this, we first define an interaction cost model that specifies the *objective value* of an interaction setting. Then, we introduce user characteristics that influence individual preferences for interaction, in terms of their objective quantities. Lastly, we define a parametric utility function that maps the objective value and user characteristics to a *subjective utility*. When computing the utility of actions to evaluate which one is best, the system uses this utility function by "plugging in" the necessary parameters based on the current state and action. Since this function defines subjective utility, the system's reasoning process chooses the action that best satisfies the user's interaction preferences.

[1]Indeed, the user modeling literature provides a suite of machine learning techniques that can be used to estimate user information quickly (e.g., see [13]).

2.2 Impact of Intelligent Actions

Different interface designs serve different purposes. Generally speaking, there are two main objectives in intelligent assistance: (i) to minimize user effort in task completions, and/or (ii) to maximize the ease of interaction during application use. In desktop applications, many kinds of system actions can be implemented to (potentially) achieve these objectives. Examples include: doing mundane work on the user's behalf (e.g., auto-completion), moving widgets to another location for more convenient selection (e.g., adding, moving, deleting widgets), changing the delivery of widgets (e.g., via animation), changing the presentation of widgets (e.g., level of transparency), sending reminders (e.g., using a text balloon), making suggestions (e.g., via a toolbar), asking questions explicitly (e.g., via a dialog box), etc. Each of these actions come with associated benefits and costs. For example, adding a frequently executed item to the menu can increase selection convenience at the risk of inducing more bloat. Among the many interaction factors proposed in the literature, we focus on the following:

- *information processing*: cost of evaluating a set of items
- *savings*: manual effort that would have otherwise been required
- *occlusion*: cost of displaying widgets in the user's workspace
- *bloat*: cost of displaying excessive functionality

For a detailed rationale of our choices, please see [12]. We present a formal model for these interaction factors in Section 2.3.

2.3 An Objective Interaction Cost Model

Since processing and savings are well-studied interaction factors, we adopt the existing quantitative models. Specifically, the cost of processing is linear for naive users [9] and logarithmic for expert users [8, 15]. To combine the two models, we define $proc = f(Expertise, Len)$, where $Expertise$ is either naive or expert, Len is the number of items to process, and f is linear for naive users and logarithmic for experts. To model savings, we adopt the GOMS-KLM model [5] that quantifies user effort in terms of the mode used to carry out an event, such as menu selection using the mouse versus keyboard shortcuts. We define the objective savings as $quality = Num * GOMS(Mode)$, where Num is the number of events and $GOMS(Mode)$ is the effort required for that mode.

Both occlusion and bloat are often mentioned in the design literature but, to our knowledge, there are no formal attempts to model them. For this reason, we conduct experiments to explore the parameters and structure of a quantitative model. As a result, we obtained $o = f(Opac, B)$ as objective occlusion, where $Opac$ is the level of opacity of an occluding widget, and B is whether the user's immediate focus is occluded. For objective bloat ("excess"), we obtained $xs = f(Unused)$, where $Unused$ is the difference between the number of functions shown and used. We refer the reader to Sections 3.1 and 3.2 for the definition of these functions and the corresponding experiments.

3. EXPERIMENTS AND SIMULATIONS

We conduct experiments to empirically derive quantitative mod-

els for occlusion and bloat. The analysis of each experiment investigates the relevance of the tested parameters and empirically derives a functional form. While occlusion and bloat have largely been neglected in experimental studies, our results show that both factors cause an interaction effect. This indicates the importance of modeling these two factors in our interaction cost model.

Both experiments had 12 volunteer participants from a graduate computer science pool, all of whom have a good command of written English and no motor control deficiencies.

3.1 Occlusion

The purpose of this experiment is to derive an objective model of occlusion in the context of intelligent assistance. We simulated a typing task by focusing on the task of typing a single letter in a sentence. Each trial consists of the user typing the highlighted letter (i.e., the target), ignoring or dismissing a pop-up box (varied in 4 parameters defined below), and then typing a second highlighted letter. We measured the time between the two typed letters in each trial. A screenshot of this program is shown in Figure 1.

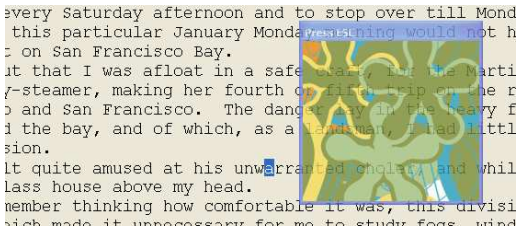

Figure 1: Screenshot showing a pop-up dialog box of size 200×200 pixels at 80% opacity in a typing task.

Each trial varied in 4 parameters of the dialog box: direction, size, opacity ($Opac$), and proximity — yielding a total of 480 configurations. In addition, we logged the intersecting area between the dialog box and the target letter. The measured task completion time in each trial is a function of these 6 variables.

To create a simpler model, we used factor analysis to determine the most important variables. We used ANOVA to determine whether each set of user data came from separate distributions, and the F-test to determine the complexity of the model. As a result, we found occlusion is best explained by a non-linear function: $o = f(B, Opac)$, where B is an indicator to denote the presence of overlap between the dialog box and the target, and $Opac$ is defined above. When $B = 0$, $o = c_0$, and when $B = 1$, the best approximation is a cubic function $o = c_3 Opac^3 + c_2 Opac^2 + c_1 Opac + c_0$ for half of the users and a linear function $o = c_1 Opac + c_0$ for the other half, where $c_0, .., c_3$ are empirically derived constants for individual users.

3.2 Bloat

The purpose of this experiment is to derive an objective model of bloat in the context of intelligent assistance. We designed a menu selection task with an interface that has the same menu structure as Microsoft Word but using abstract menu labels. This experiment has 4 conditions varying in the number of menu items shown ($Shown$): 18, 62, 107, and 152, out of a total of 152 possible menu items. In all the conditions, we fixed the number of menu items used ($Used$) to 15. The target items in the selection task are randomized across conditions. In each trial, participants follow an instruction (e.g., Fruits → Papaya) and select the target menu item. We measured the successful selection time of target menu items. A screenshot of the program is shown in Figure 2.

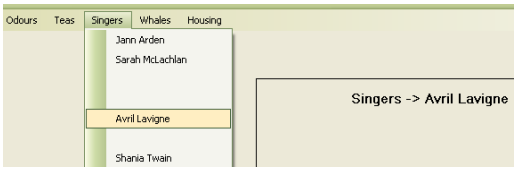

Figure 2: Screenshot showing the target menu item and instructions on the right. Notice this menu has many empty slots.

Each participant carried out 15 trials per condition. With 4 conditions, each participant carried out a total of 60 trials in the experiment. We counterbalanced the order of blocks using a size 4 Latin square. The measured selection time in each trial is a function of $Shown$ and $Used$. Conceptually, we defined $Unused$ as the number of items shown but not used. Using this definition, we used ANOVA and an F-test and found that bloat is best approximated as a linear function $xs = c_1 Unused + c_0$ for most users, and as a quadratic function $xs = c_2 Unused^2 + c_1 Unused + c_0$ for 1 user, and as a cubic function $xs = c_3 Unused^3 + c_2 Unused^2 + c_1 Unused + c_0$ for 1 user, where $c_0, .., c_3$ are empirically derived constants for individual users.

3.3 Markov Decision Process

To put the interaction cost model to use, we designed a system that adapts menus in simulation. The first step in the design is to identify the relevant interaction factors for this domain. Given the objective cost models of these factors (as defined in Section 2.3), we introduce user characteristics and define the overall subjective utility function. This function is used in the system to evaluate the goodness of its actions. In the simulation, we assume we know the user characteristics and model the customization problem as an MDP[2]. In this way, the agent optimizes its adaptive actions with respect to the user's preferences over repeated interaction with the system. When an MDP is solved, we obtain a *policy* that maps (application and user) states to an optimal action. For a detailed introduction to MDPs, the reader is referred to [4].

The possible actions of this system are to add a menu item, delete a menu item, or do nothing. For simplicity, we use bloat and savings in defining this system's interaction cost model[3]. These two factors are relevant because the system can remove or introduce items that offer potential savings. To compute the subjective utility of system actions involving these factors, we use the following functions:[4] $savings = f(Quality, N, D, F, I)$ — represents the perceived savings of the resulting interface, given the objective quality of savings, how much help the user needs (N), how distracted the user is in general (D), how frustrated the user is with the system (F), and whether the user generally likes to work independently (I); $bloat = f(XS, T, D)$ — represents the perceived bloat of the resulting interface, given the objective excess of bloat, the user's tolerance toward bloat (T denotes whether users are *feature-keen* or *feature-shy* [16]), and how distracted the user is with more functions available (D). Finally, the overall utility of an action is the weighted sum of $savings$ and $bloat$.

[2]In reality, this problem should be modelled as a *partially* observable MDP because we cannot know the user with complete certainty. However, since machine learning techniques are available for learning the user, we assume we can observe the user here.

[3]In general, the cost of processing, interruption, and disruption also play a role in adaptive menus.

[4]Due to a lack of space, we refer readers for a detailed account of these models elsewhere [13, 12].

In the simulation, we discretize the user variables to be binary and tertiary (e.g., F has 3 values, representing the user being highly, somewhat, and not at all frustrated). With 5 user variables, the system's utility function accounts for a total of 162 user types. In addition, the MDP dynamics are defined to reflect changes to the interface (adding/deleting) can distract and frustrate the user. In this way, the system does not risk taking adaptive actions when dealing with highly distracted/frustrated users.

We conducted two simulation runs. The first one investigates the effect of bloat and the second one explores the system's adaptability toward various user types in the model. At any point in time, the adaptive menu can have 1 to 6 items shown, and the system policy suggests to add an item, delete an item, or do nothing, based on the menu state and the user's type. In the first simulation, we defined a constant value for savings and focus only on bloat. A qualitative description of the results is presented in Table 1, where the number of menu items shown is categorized as "few" (less than half), "many" (more than half), or "any" (any value between 1 to 6).

Distractibility	Tolerance	Shown	Policy
low/medium	keen	any	add
high	keen	few	add
low	shy	many	delete

Table 1: Results showing the effect of bloat.

Generally, we see that the system adds items for feature-keen users, even when they are highly distracted because an addition offers enough savings to tradeoff the cost of annoying the user. For all other combinations, the system opts to do nothing.

In the second simulation, we re-introduced savings to compare the adaptive behaviour toward different user types. When the user's frustration and independence levels are low and the neediness level is high, we expect this type of user to be most receptive to help. We define this user type as our "best case". Analogously, we define the "worst case". The qualitative results are shown in Table 2.

Case	Distractibility	Tolerance	Shown	Policy
best case	low	keen/shy	any	add
best case	medium/high	keen	any	add
worst case	low	keen	any	add
worst case	low	shy	many	delete
worst case	medium	shy	many	delete

Table 2: Results showing the effect of user types.

For the best case user type, the system tends to suggest adding an item because these users are receptive to adaptive help. For the worst case user type, the system is much more conservative and only adds an item for low distractibility and feature-keen users. The system deletes items when many are shown for feature-shy users who are not highly distractible. For all other combinations, the system opts to do nothing in fear of distracting the user by changing the interface. Note that there are 160 user types "between" the best and worst cases. These results show that the system is able to adapt to many different user types.

4. CONCLUSIONS

In this paper, we proposed a decision-theoretic approach to account for the varying user preferences with software interfaces. We modeled four interaction factors, that when combined, result in an interaction cost model that forms part of a utility function used to explain different interaction preferences. Our interaction cost model is highly flexible so that formal models can be refined simply by changing the corresponding formula in the model. Additionally, our implementation shows that designers can pick and choose the interaction concepts from the framework that are relevant for their application. By modeling the costs and benefits of various interaction factors, intelligent systems can reason the impact of its actions and optimize its behaviour for different users with varying interaction preferences.

5. REFERENCES

[1] D. Albrecht, I. Zukerman, and A. Nicholson. Pre-sending Documents on the WWW: A Comparative Study. In *Proc. of IJCAI*, pp. 1274–1279, 1999.

[2] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *Proc. of IJCAI*, pp. 1293–1299, 2005.

[3] T. Bohnenberger and A. Jameson. When policies are better than plans: decision-theoretic planning of recommendation sequences. In *Proc. of IUI*, pp. 21–24, 2001.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94, 1999.

[5] S. Card, P. Moran, and A. Newell. *Psychology of HCI*. Hillsdale, NJ: Erlbaum, 1980.

[6] C. Conati, A. Gertner, and K. VanLehn. Using Bayesian networks to manage uncertainty in student modeling. *Journal of UMUAI*, 12(4):371–417, 2002.

[7] K. Gajos, M. Czerwinski, D. Tan, and D. Weld. Exploring the Design Space For Adaptive Graphical User Interfaces. In *Proc. of AVI*, pp. 201–208, 2006.

[8] W. Hick. On the rate of gain of information. *Journal of Experimental Psych.*, 4:11–36, 1952.

[9] A. Hornof and D. Kieras. Cognitive modeling reveals menu search is both random and systematic. In *Proc. of CHI*, pp. 107–114, 1997.

[10] E. Horvitz. Principles of mixed-initiative. In *Proc. of CHI*, pp. 159–166, 1999.

[11] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proc. of UAI*, pp. 256–265, 1998.

[12] B. Hui. A Survey of Interaction Phenomena. Technical report, Dept. of Computer Science, Univ. of Toronto, 2007.

[13] B. Hui and C. Boutilier. Who's Asking for Help? A Bayesian Approach to Intelligent Assistance. In *Proc. of IUI*, pp. 186–193, 2006.

[14] B. Hui, S. Liaskos, and J. Mylopoulos. Requirements Analysis for Customizable Software. In *Proc. of RE*, pp. 117–126, 2003.

[15] R. Hyman. Stimulus information as a determinant of reaction time. *Journal of Experimental Psych.*, 45:188–196, 1953.

[16] J. McGrenere, R. Baecker, and K. Booth. An evaluation of a multiple interface design solution for bloated software. In *Proc. of CHI*, pp. 163–170, 2002.

[17] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, NY, 1994.