A Probabilistic Mental Model for Estimating Disruption

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

Grant Partridge
Dept. of Computer Science
University of Manitoba
umpartridge@cs.umanitoba.ca

Craig Boutilier
Dept. of Computer Science
University of Toronto
cebly@cs.toronto.edu

ABSTRACT

Adaptive software systems are intended to modify their appearance, performance or functionality to the needs and preferences of different users. A key bottleneck in building effective adaptive systems is accounting for the cost of *disruption* to a user's *mental model* of the application caused by the system's adaptive behaviour. In this work, we propose a probabilistic approach to modeling the cost of disruption. This allows an adaptive system to tradeoff disruption cost with expected savings (or other benefits) induced by a potential adaptation in a principled, decision-theoretic fashion. We conducted two experiments with 48 participants to learn model parameters in an adaptive menu selection environment. We demonstrate the utility of our approach in simulation and usability studies. Usability results with 8 participants suggest that our approach is competitive with split menus w.r.t. task performance, while providing the ability to reduce disruption and adapt to user preferences.

ACM Classification Keywords

I.2 Artificial Intelligence: Misc.; H.5 Information Interfaces and Presentation: Misc.

Author Keywords

Disruption, probabilistic mental model, decision-theoretic systems, user modeling

1. INTRODUCTION

The need to develop user adaptive systems has been made evident by emerging work that applies user modeling to technologies ranging from automatic interface customization [7, 11] to health care systems [1]. Since different people prefer different styles of interaction, intelligent systems should adapt to individual user's changing needs and preferences. For example, some users may prefer to optimize task performance, while others may prefer to learn the software better at the expense of task completion time. The sequential nature of human-computer interaction (HCI) makes it possible to build user adaptive systems that learn user preferences over time. When deployed successfully, adaptive systems have the potential to increase productivity and user satisfaction.

Through repeated interaction, a user builds a *mental model* of the application that reflects the knowledge gained through experience. This may include available software functionality, the locations of functions, the effects of those functions (e.g., how long they take, whether multiple actions achieve the same result), etc. A key bottleneck in building effective adaptive systems is accounting for the induced *disruption* to a user's mental model. Consider for instance menu selection. The first time a user selects `Exit` from the `File` menu, he scans all the items inside `File`. After using `Exit` several times, he learns it is located at the bottom of `File`. An adaptive system may observe `Exit` is frequently used, and decide to move it to the top of `File` so that future access becomes faster. Obviously, there are tradeoffs involved: there are long-term task performance gains (the user will access the frequently used `Exit` more quickly); however, the disruption of the user's mental model of `Exit`'s location may cause short-term performance degradation—more search is involved until the new location is learned—and annoyance. The degree of disruption is, intuitively, related to the strength of the user's prior beliefs and the degree of "new search" required (e.g., if the new location is near the old one, disruption may be less than if it were further away).

While notification mechanisms may ease the abrupt transition caused by adaptive actions, some users may find them distracting or unnecessary. Ideally, an adaptive system should assess a user's mental model, and make the tradeoffs between the *long-term* benefits of adaptive actions (e.g., improved task performance) and the costs of disruption before taking those actions. We propose just such a model here, with a focus on models of function location, relevant to adaptive systems that change function locations in order to make access more convenient or to reduce interface bloat. Our mental model is probabilistic: it allows for a natural definition of strength, model dynamics (including learning and forgetting), and cost of disruption. We also propose means for assessing the long-term tradeoffs in a decision-theoretically principled fashion. This stands in contrast to adaptive system models that focus only on maximizing benefits of speed performance, while designating a "generic" cost to adaptive actions or ignoring costs altogether [12].

In Sec. 2, we review some known properties about mental models from existing literature. As our first contribution, we present a probabilistic mental model in Sec. 3 and define three mental operations (learning, forgetting, disruption) using mental model strength. In Sec. 4, we describe how our model is used to estimate the cost of disruption in a decision-theoretic system. Unlike other user adaptive

CHI 2009, April 3 - 9, 2009, Boston, MA, USA.

systems, ours is the first to explicitly tradeoff the long-term benefits of adaptive actions against the associated cost of disruption. Our system is first evaluated in simulation in the context of adaptive menus in Sec. 5. To learn a more realistic model, we conducted two studies reported in Sec. 6 and Sec. 7. Most importantly, these experiments show a significant overhead in search performance times that varies with mental model strength and cannot be naturally explained by existing predictive models. Finally, we confirm our simulation results by conducting usability experiments described in Sec. 8. Both the objective and subjective usability results show that our approach offers competitive performance with adaptive menus, while enabling more opportunities for users to learn the system.

2. PROPERTIES OF MENTAL MODELS

The term *mental model* has been used in psychology to denote a person's mental representation of (concrete or abstract) objects [13, 8]. We use the term in its HCI sense, pertaining to software usability—a mental model is a user's representation of an application. In particular, we focus on mental models of function locations, e.g., the location of menu items. Much research on mental models describes conditions and effects, but does not offer explanations or theoretical predictions [16]. Furthermore, due to variations in the elicitation and experimental procedures, much debate surrounds these results [2].

Despite the imprecise state of work in the area, most researchers agree that people's mental models are *incomplete*, *dynamic*, and *unstable* [14]. Consider a word processing application with standard menus like `File`, `Edit`, etc. A typical user will learn where common functions are located within menus, but will not know the locations of all available functionality, thus rendering his mental model incomplete. Through extended usage, the user reinforces, or strengthens, what he already knows and learns new information about the application. Certain knowledge may weaken as well (e.g., rarely used functions). This illustrates the dynamic nature of a mental model. Finally, if we ask the user to report his beliefs (based on his current mental model), elicited responses will often vary, even without disruption to the mental model, indicating instability of mental models.

In HCI, mental models are often used to account for software *learnability*, or the user's ability to master the application. For example, after usage, a researcher may elicit/evaluate how well a user's mental model depicts the actual state of the software. If a system could "track" a user's mental model at runtime, it could identify sources of disturbance to the mental model and eliminate them from the design of the software. To this end, some computational approaches to mental models have been proposed, but are non-probabilistic and use heuristic updates [17]. Thus, these approaches do not account for the three properties above in a principled manner.

Our aim is to develop an approach to mental models that allows a system to quantify the disruption caused by adaptive actions. We are unaware of any such formal models of disruption. In Sec. 3, we present a probabilistic representation that adequately captures the three mental model properties.To make our discussion more concrete, we consider three types of adaptive actions in menu selection tasks (although the general principles apply more broadly): moving a function to the top and shifting functions down as needed (TOP); swapping a function with the one above it (SWAP); hiding a function at the bottom of a menu (e.g., under double arrows) and shifting functions as needed (HIDE); and the non-adaptive default of not moving any items (NONE). We aim to capture the effect of such actions on a user's mental model in order to quantify the induced disruption.

3. A MENTAL MODEL OF FUNCTION LOCATION

Most mental model research elicits information to determine the representation of mental models, "rather than trying to determine which set of hypothetical mental operations is supported by the data" [16]. Here, we develop a probabilistic representation of a mental model for function location that supports three key operations of interest for adaptive systems: learning, forgetting, and modeling disruption.

3.1 Basic Model and Model Strength

Let K be a set of possible functions each located in one of L (menu or interface) locations. For any $k \in K$, the user's mental model of k's location, θ^k, is a multinomial distribution over L locations. We might think of $\theta^k(l)$ as the probability the user will attempt to (first) access function k at location l. While generally not true, we assume for simplicity that the distributions $\theta^1, \ldots, \theta^K$ are independent, allowing a convenient marginal (rather than joint) representation.

Without experience, model θ^k will be "weak" (e.g., uniform over L or a prior induced from using similar interfaces). Using k then generally requires the style of search typical of novices (e.g., visual search) [10]. After executing k some number of times, we expect the model θ^k to become stronger (and more accurate), with higher probability assigned to the true location (and possibly nearby locations, spatially or analogically), leading to reduced search time, and eventually leading to expert behavior (e.g., logarithmic search or "immediate" execution) [3]. Indeed, *strength* of a mental model, or how well users believe they know a function's location, seems to be a key criterion in how they assess their own level of interface expertise. Our model leads to a natural notion of strength, which we use extensively below to quantify disruption. In what follows, we focus our formalization on model strength, rather than the model itself.

Model strength is defined w.r.t. to a user's degree of uncertainty or normalized entropy of the model distribution. Specifically, the strength of θ^k is $M^k = 1 - H(\theta^k)/H_L^+$, where $H(X) = -\sum_x P(x)\log(P(x))$ is the entropy of distribution X, and H_L^+ is the maximum entropy of any multinomial with L events (i.e., the entropy of the uniform distribution over L locations). This definition normalizes strength in $[0, 1]$, with 0 corresponding to the "weakest" mental model (high normalized entropy, no knowledge of function location) and 1 corresponding to the "strongest" mental model (low normalized entropy, complete certainty of location).

Using this representation, we develop the dynamics of mental models, with an eye toward quantifying disruption cost, using three key operations: learning, forgetting, and disruption due to adaptation. (The first two apply to any form of software, adaptive or not.)

3.2 Learning

We begin by examining how location models are learned. As noted, we expect model θ^k to become stronger with increased usage of k. Additional factors may also influence this process. For example, using a function k' located *near* k may reinforce θ^k somewhat—searching for k' may involve seeing (and learning) k's location.[1] Thus, usage history of k and neighbouring functions can influence θ^k. Other factors are visual-spatial cues surrounding k. For example, menu length and depth may influence how well users know menu item locations (longer/deeper menus leave greater possibility for error). Landmarks could accelerate learning, with functions near landmarks—line separators, submenu arrows, disabled (greyed-out) functions, shortcut labels, top or bottom of a (sub)menu—learned more rapidly.

Abstractly, let $Context$ denote the set of function usage histories and cues that influence the mental model of k. The learning process can be encoded as $\theta^k = f'(Context)$. As discussed above, varying contexts afford different learning rates, as illustrated conceptually in Fig. 1. In terms of the *dynamics* of model strength, we assume that strength M_t^k at time t is a linear combination of the prior strength and strength induced with the new usage context C_t^k:

$$M_t^k = (1-\lambda)M_{t-1}^k + \lambda C_t^k \quad (1)$$

Here, $\lambda \in [0,1]$ is a learning rate and $C_t^k = f(Context_t)$ is the strength associated with $\theta_t^k = f'(Context_t)$, but ignores the dynamics of the update process. By incorporating the prior strength, this definition is consistent with the concept of *belief perseverance* [15], whereby people maintain strong beliefs in the face of contrary evidence.

We investigate the relationship between the user's mental model and several cues and usage histories empirically in Sec. 6. From this, we derived $C_t^k = f(Freq_t^k, NB_t^k)$, where $Freq^k$ is the accumulated usage frequency of k and NB^k is the usage frequency of k's neighbours. Specifically, $C_t^k = a^f \cdot \log(b^f \cdot Freq_t^k) + c^f$, with $a_h^f = 0.11$, $b_h^f = 0.51$, $c_h^f = 0.56$ when NB_t^k is high, and $a_h^f = 0.11$, $b_h^f = 0.44$, $c_h^f = 0.50$ when NB_t^k is low. We found little evidence of a learning effect with cues (see Sec. 6 for details).

(a) Faster rate of learning.

(b) Slower rate of learning.

Figure 1. Learning and changes in the mental model distribution.

[1] This is one reason our assumed model independence fails to hold.

3.3 Forgetting

When functions are not used, we expect users to forget their locations over time. This implies a decrease in model strength (and accuracy in most cases), and an increase in search time. Fig. 2 illustrates this process, with an unused function becoming more uncertain.

Figure 2. Forgetting and changes in the mental model distribution.

We model the rate of decrease in strength using exponential decay, as suggested by other memory literature [4]. The impact of disuse on model strength is given by:

$$M_t^k = \beta M_{t-1}^k \quad (2)$$

where k is unused at time t, and $0 < \beta < 1$ is the forgetting rate (w.r.t. model strength, as distinct from decay of model parameters themselves).

Using the processes modeled in Eqs. (1) and (2), we tracked model strengths in simulation in the context of menu selection. In a menu with $K = 20$ items and $L = 20$ locations, the user repeatedly selects a series of items according to a Zipf distribution [19]. Fig. 3 shows the resulting dynamics. In general, used functions exhibit logarithmic growth in strength while unused functions exhibit exponential decay.

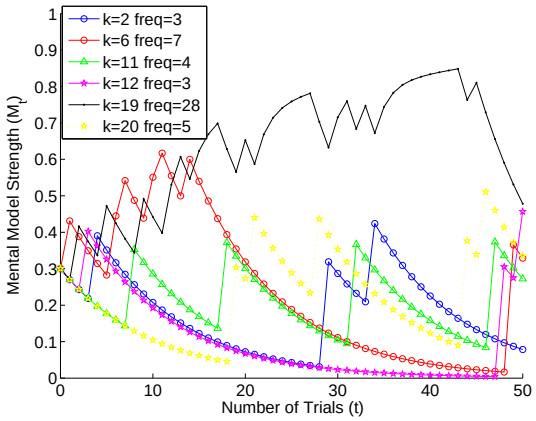

Figure 3. Dynamics of strength estimates over 50 trials with $M_0^k = 0.3$ for all k. Strengths of several functions (indexed as $k = 2, 6,$ etc.) with varying usage frequencies (of 50 trials) are shown.

3.4 Model Disruption

Modeling the effect of adaptive actions on mental models, and strengths, is critical to assessing their utility. When k is moved to a new location, intuitively, we expect θ^k to change in a way that reflects the learning process in the new context (function arrangement) and the previous one. This is due to our expectation that users will retain some memory of k's previous location, even after realizing it has been moved (an effect like "muscle memory").

Formally, we capture this effect as a mixture of two models. Let ϕ^k denote a hypothetical model learned (using the learning and forgetting processes above) as if the user had no experience with k prior to the adaptation that moved it. The user's model at time t following the adaptation is given by the mixture $\theta_t^k = (1-\alpha)\theta_{t-1}^k + \alpha\phi_{t-1}^k$, where α denotes

a learning rate (the rate at which the new model replaces the old one). The process is illustrated in Fig. 4 where k is moved to a new location after $t = 3$. Notice that the "new" model (which is mixed with the old) also evolves over time as the user gains experience with the "new" location.

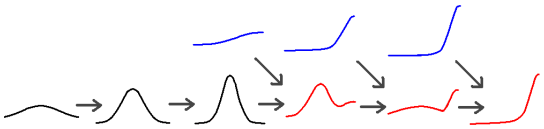

Figure 4. Changes in θ^k as k is moved to a new location at time $t = 3$. The bottom distribution represents θ^k and the consequences of moving k. The top distribution represents the hypothetical mental model ϕ^k of learning k in a new context. Prior to $t = 3$, the dynamics of θ^k following the regular learning and forgetting processes. Once k is moved, the resulting distribution is $\theta^k_t = (1-\alpha)\theta^k_{t-1} + \alpha\phi^k_{t-1}$.

In what follows, we model impact of disruption *on model strength* M^k instead of the model θ^k itself. This approximation is more tractable (requiring tracking strength only, not models). We assume that $M^k_t = (1-\alpha)M^k_{t-1} + \alpha M(\phi^k_{t-1})$. (The strength learning rate α will not be the same as the model learning rate, but can be derived from it.)

Adaptive actions disrupt a user's mental model and incur two types of costs. First, referring to Fig. 4, we see a location change initially has a negative impact on task performance or search behavior, due to the reduction is model strength and accuracy. At $t = 3$, θ^k is peaked and user can assess k quickly. Immediately after the move, the user will search for k at the old location, not find it, and need to discover the new location (e.g., using visual linear search). This incurs an (objective) *disruption time*. In addition, the user probably experiences some level of frustration (due to both mental effort and increased task time), which is reflected in a subjective *annoyance* factor. In Sec. 7, we conduct an experiment to learn the objective disruption time. In practice, the annoyance factor may amplify the disruption time so that users who dislike adaptations perceive a more significant cost. We do not address annoyance further here. Notice, of course, that disruption time decreases over time as the new model is learned. We discuss this further in the next section.

4. DECISION-THEORETIC ACTION SELECTION

A typical metric for evaluating adaptive systems is the degree to which they reduce user effort in repetitive tasks (e.g., typing common phrases, or selecting oft-used menu items or toolbar icons). The costs associated with adaptations are often ignored. Here, we develop a decision-theoretic model that allows benefits (e.g., selection time savings) and costs (e.g., selection time increases due to disruption) to be traded off against one another. Our model is also sequential, reflecting tradeoffs over time.

4.1 The Benefits of Savings

An adaptive action provides savings for the user if it moves k to a location that is closer to the user's reach. We adopt Fitts' Law [5] and define the *savings* of moving k from l^k_{t-1} to l^k_t as $S(l^k_{t-1}, l^k_t) = \mathit{fitts}(l^k_{t-1}) - \mathit{fitts}(l^k_t)$, where $\mathit{fitts}(d) = a\log_2(d/w + 1) + b$ with w the width of a menu item, d the distance of the item from the parent menu item, and a and b are empirical constants.[2]

The savings *expected* of any (single) interaction depend on the probability of function usage. Moreover, functions cannot (usually) be moved in isolation (e.g., others may shift). Thus, we define the *joint expected savings* (JES) of a vector of location moves to be:

$$JES(l^{1:K}_{t-1}, l^{1:K}_t) = \sum_{k=1}^{K} p^k S(l^k_{t-1}, l^k_t) \quad (3)$$

where $1{:}K$ is shorthand notation for K variables, p^k is k's estimated probability (e.g., based on sample frequency). Since an adaptive action A fully determines $l^{1:K}_t$ given $l^{1:K}_{t-1}$, we also denote this by $JES(A|l^{1:K}_{t-1})$.

4.2 The Costs of Disruption

When an adaptive action occurs, we define disruption time (see Sec. 3.4) as the search time required for the function selection task over the time required given the prior user model θ^k. Since this is an objective measure, we devised an experiment in Sec. 7 to learn this value. With adaptive action A being one of NONE, SWAP, TOP, and HIDE, we define (initial) disruption time as $D^k_t(A) = g(A|M^k_t)$, where g takes the form $a^d M^k_t + b^d$, with empirical constants of $a^d_n = 0, b^d_n = 0$ for NONE, $a^d_s = 1.4, b^d_s = 0.2$ for SWAP, $a^d_t = 2.8, b^d_t = 0$ for TOP, and $a^d_h = 6.0, b^d_h = 2.9$ for HIDE. Using this definition, we define *joint expected disruption* (JED) analogously to JES:

$$JED(A|M^{1:K}_t) = \sum_{k=1}^{K} p^k D^k_t(A) \quad (4)$$

Eq. (1) gives us a way to estimate the dynamics of model strength with function use. When function k is moved as a result of adaptive action A, strength becomes weaker. We capture this with a disruption factor δ, which weakens the model after applying A:

$$M^k_t = \delta M^k_t \quad (5)$$

The factor by which strength weakens is defined as a function of disruption time $D^k_t(A)$. Specifically, given M^k_t, we expect no change in strength when disruption time $D^k_t(A) = 0$ (i.e., A is NONE). In that case, $\delta = 1$. When $D^k_t(A)$ is small, users may not notice the disruption, so we expect strength to only decrease slightly. As $D^k_t(A)$ increases and becomes noticeable, we expect strength to decrease. The greater disruption time, the less the effect on strength because the mental model will, in the worst case, "reset" itself to the weakest point. The specific function we crafted to model this pattern is a Gaussian function defined as $\delta = exp^{-(D^k_t(A)/c)^2}$, with $c = 3.977$. In effect, there is minimal impact when disruption time is between 0 to 1.5 second. When $D^k_t(A) > 1.5s$, the impact peaks and eventually stabilizes beyond $D^k_t(A) > 6s$. These boundaries are developed using the empirical results from Sec. 7 as guidelines.

4.3 Sequential Decision Model

The savings and disruption cost models above focus on a single interaction. In repeated use, both savings and disruption

[2]We require an application-specific mapping from locations to distances, which reflects specific implementation artifacts (e.g., to access a hidden menu item, hovering or clicking double arrows first).

costs accrue over multiple interactions. However, disruption cost diminishes over time as the user develops a new mental model as described in Sec. 3.4. The utility of an adaptive action must take into account the sequential nature of the interaction and total net utility over some horizon of interest. For simplicity, we model the long term effects of the current adaptation assuming no future adaptive actions are taken.

Let $\mathcal{H}$ denote the expected number of interactions with an interface. We assume discount factor $0 \leq \gamma \leq 1$, with long-term savings given by: $\sum_{h=1}^{\mathcal{H}} \gamma^h JES(A|l_{t-1}^{1:K})$. Long-terms costs are defined similarly, but with a wrinkle. Unlike savings, the long-term effects of disruption decay over time, as the user learns the new model at rate α. With the old model's retention rate as $1-\alpha$, we discount *JED* at rate $1-\alpha$ (in addition to γ): $\sum_{h=1}^{\mathcal{H}} (1-\alpha)^h \gamma^h JED(A|M_t^{1:K})$.

While one could sum the two expressions above to obtain the estimated utility of an adaptive action, we wish to account for different user preferences. Specifically, we imagine that some users have a strong preference for interfaces that support fast selection (and more generally, task completion) time, even if they are somewhat "disruptive"; others may prefer a more stable interface, even at the expense of increased expected selection time. This is accomplished heuristically by scoring actions using a convex combination of the scores above, with weight $w_s \in [0,1]$ applied to *JES* and $w_d = 1 - w_s$ applied to *JED*. Users with greater w_s are more tolerant of disruption if this induces selection savings. This gives the *weighted expected reward* score, $WER(A|M_t^{1:K}, l_{t-1}^{1:K}, w_s, \mathcal{H}, \alpha, \gamma)$:

$$\sum_{h=1}^{\mathcal{H}} \Big[w_s \gamma^h JES(A|l_{t-1}^{1:K}) - w_d (1-\alpha)^h \gamma^h JED(A|M_t^{1:K}) \Big] \quad (6)$$

To compute the optimal action, the WER system policy is:

$$A^* = \arg\max_A WER(A|M_t^{1:K}, l_{t-1}^{1:K}, w_s, \mathcal{H}, \alpha, \gamma) \quad (7)$$

Since we only consider specific kinds of adaptive actions (i.e., NONE, SWAP, TOP, BOTTOM), only a few location changes are possible (specifically, the neighbouring, top, and bottom locations of l^k).

5. SIMULATION RESULTS

To illustrate the benefits of our model, we conducted a simulation experiment to compare the performance of several system policies in the context of adaptive menus. We report analogous usability results with real users in Sec. 8. The scenario is repeated interaction focusing on menu selection tasks. In each session, a (simulated) user must select an item, drawn from a predefined item distribution. $\mathcal{H}$ items are drawn in turn and the system can adapt its menu according to some policy after each interaction.

Policies: We compare seven menu policies w.r.t. selection and disruption. BEST STATIC provides an upper-bound on performance: it presents the menu layout with items sorted in descending frequency. Aside from BEST STATIC, no other policy has access to the item distribution. At the opposite end of the spectrum, we tested the adaptive policy RANDOM-N, which randomly moves N items at each interaction (it is maximally disruptive). The remaining policies make use of *estimated* item distributions to adapt their menus.

Split menus have been shown to offer faster selection performance than various other static and adaptive menus [18]. A split menu consists of two areas—the top (adaptive) partition has N menu items with highest estimated usage (in order), while the bottom (static) partition has $K-N$ remaining items arranged in a fixed (default) ordering Setting $N = 4$ is understood as providing good performance in practice [18]. We refer to this policy as SPLIT-4.[3].

Our adaptive policy is denoted WER(w_s)-N, and is parameterized by the savings weight (and hence, disruption weight) and the number of items moved simultaneously (N). We implement action selection in Eq. (7) greedily (the best move is derived first, then the next best *given* the first, etc.). We test three versions by varying $w_s = 0.1, 0.5, 0.9$. Finally, we test the JES-N policy, maximizing joint expected savings from Eq. (3). This alternative adaptive approach ignores disruption (and hence need not reason sequentially).

We investigate these policies under two item distributions: Zipf (which models actual function usage frequencies [9]) and uniform. We fix the number of menu items to 20 ($K = 20$ and $L = 20$), the learning rates to $\lambda = 0.5$ and $\alpha = 0.5$, the forgetting rate to $\beta = 0.9$, the savings discount factor to $\gamma = 0.95$, and the horizon to $\mathcal{H} = 50$. To enable comparative results with SPLIT-4, all the adaptive systems take TOP actions only with $N = 4$. All the systems estimate frequency using normalized sample frequency.

Simulated Users: The simulated users select menu items by sampling from the predefined item distribution. To model selection times, we adopt the model of Cockburn et al. [3] which accounts for novice and expert performance (which is necessary in adaptive systems since interface changes can lower a user's degree of expertise). In this model, item selection requires the user to (i) mentally decide to select an item and (ii) physically select it. Selection time for item k is $T^k = T_d^k + T_p^k$ (i.e., decision time plus pointing time). Pointing time is roughly the same for everyone (using Fitts' law). However, decision time varies with *expertise* w.r.t. k's location. Decision time is modeled as a linear combination of novice and expert search times with $\epsilon^k \in [0,1]$ denoting the user's expertise with k [3]:

$$T_d^k = (1-\epsilon^k) T_{vs} + \epsilon^k T_{hh}^k \quad (8)$$

where T_{vs} and T_{hh} are previously proposed search models for novices and experts respectively (see [3] for details).

To estimate the user's expertise with an item location, we equate $\epsilon^k = M_t^k$ based on our mental model strength estimate. In effect, when $M_t^k = 1$ (strong), the user is a complete expert with $\epsilon^k = 1$. As M_t^k decreases (becomes weaker), ϵ^k approaches to the value of a novice.

Results: We define *selection time* as the time predicted according to [3], and *disruption time* as any additional search time (Sec. 4.2.3). To assess the impact of adaptation on learnability, we measure how often a user develops a "strong" mental model of any functions in the session (*total strong models*). We deem a model to be strong when $M_t^k > 0.4$,

[3]Refinements of split menu design have been proposed recently, e.g., the "copy" variant of [6], which we discuss further below.

Method	N	Select. Time	Disrupt. Time	Total Strong Models	Percent Strong Moves
BEST STATIC	0	1197	0	152	0.00
RANDOM	4	1306	196	136	56.80
SPLIT	4	1213	22	150	3.11
JES	4	1301	130	133	36.31
WER(.1)	4	1296	0	152	0.00
WER(.5)	4	1284	6	152	0.18
WER(.9)	4	1273	50	149	2.58

Table 1. Results using a Zipf distribution with $K = 20$ menu items, averaged over 100 runs, each with a horizon $\mathcal{H} = 50$. Times in msec.

Method	N	Select. Time	Disrupt. Time	Total Strong Models	Percent Strong Moves
BEST STATIC	0	1700	0	100	0.00
RANDOM	4	1701	120	93	49.61
SPLIT	4	1702	44	95	7.33
JES	4	1703	82	91	45.60
WER(.1)	4	1700	0	100	0.00
WER(.5)	4	1700	2	100	0.10
WER(.9)	4	1702	28	97	4.42

Table 2. Results using a uniform distribution with $K = 20$ menu items, averaged over 100 runs, each with a horizon $\mathcal{H} = 50$. Times in msec.

a threshold based on the sample median of strengths from experiments in Sec. 6. We also measure how often a strong mental model is disrupted. We define a *strong move* to be any action that moves a function with a strong model to a new location, and record the percentage of strong models that are moved (*percentage of strong moves*). Ultimately, we are interested in systems that can tradeoff reduction in selection time (via adaptation) and stability (by minimizing disruption of strong models, where costs are greatest).

The results using a Zipf distribution is presented in Table 1. We report the average times based on 100 simulation runs. Note that computation times are fast in all cases ($< 50ms$). BEST STATIC is the gold standard for desirable selection time and stability. SPLIT-4 is fast, as is evident from the literature. All WER variants perform about the same, about 60-83ms slower than SPLIT-4 and 5-28ms faster than JES-4. RANDOM-4 is a lower baseline and induces an undesirable amount of disruption. As expected, JES-4 ignores disruption so it has a cost close to that of RANDOM-4. The remaining methods induce a similar and much smaller range of disruption times. Although these times are small, the associated, subjective annoyance factor of adaptation should play a role in amplifying overall costs. As a result, we expect task completion times in practice to be greater than the sum of the predicted selection and disruption times (we see this in the usability results as well). Thus, the ability to accommodate user preference toward adaptation is crucial.

When the WER weight setting is $w_s = 0.1$ (i.e., representing a user who cares most about minimizing disruption), no disruption is induced. Indeed, our data indicates this policy behaves like a static system so as to not annoy this type of user. Only when setting $w_s = 0.9$ (i.e., a user who prefers to maximize selection performance) is when we see more disruption than SPLIT-4.

The more a system adapts its interface, the less the user is able to learn and develop a strong mental model of it. We see that the number of total strong models that RANDOM-4 offers is much less than that of BEST STATIC. Since WER(.1)-4 behaves like a static system, it also offers just as many opportunities to develop strong mental models as BEST STATIC. We see a similar pattern among these systems when comparing the percentage of strong moves made. When comparing how other systems perform with respect to the percentage of strong moves made, we see that all the WER variants make fewer disturbances than SPLIT-4.

Results using a uniform frequency distribution are presented in Table 2. Not surprisingly, selection time is difficult to optimize regardless of the policy; users have fewer opportunities to develop strong models (since experience is distributed over more items). As a result, selection times are slower and strong models are rarer. The performance across other dimensions is similar for all policies, with the notable exception that all WER variants outperform SPLIT-4 in all dimensions. Overall, our WER policy is competitive with SPLIT-4 and is able to adapt to user preferences.

6. LEARNING MENTAL MODEL DYNAMICS

Our first experiment is designed to determine the kinds of visual-spatial cues and usage histories that define the learning context for model strength. First, participants were faced with a series of controlled menu selection tasks. This allowed participants to build a mental model of our experimental interface. Second, the same participants were presented with a series of recall tasks designed to assess their model strengths. In total, we collected data from 48 participants.

6.1 Training Session

The training session was designed to accommodate a range of cues and usage conditions in order to test the effect of these variables on the mental model. Participants were asked to complete a set of 29 menu selection tasks as accurately and as quickly as possible using abstract labels in a pull-down menu as shown in Fig. 5. Only four distinct (randomly chosen) menu items were used as targets among the 29 tasks. If a mistake was made, the participant had to redo the task. This set of tasks is designed to mimic a particular *history* of application use, so we can probe the participant's mental model (in the next session) knowing the usage history that "created" it. In each trial, we measured task completion time and accuracy. We used these results to filter out participants who did not take the experiment seriously.

While we recognize the potential impact of all of the variables discussed in Sec. 3.2, we restricted our attention to two cues and two usage variables:

- *Length*: Total number of menu items (shown and hidden) belonging to the menu. Values: K=20, 40.
- *Freq*: Target usage frequency. One usage frequency is assigned to each of the four distinct targets. Values: 1, 4, 8, 16. These give the 29 tasks.
- *Landmark*: A line separator in the menu. Values: none, line above the item with $Freq = 16$, line above the item

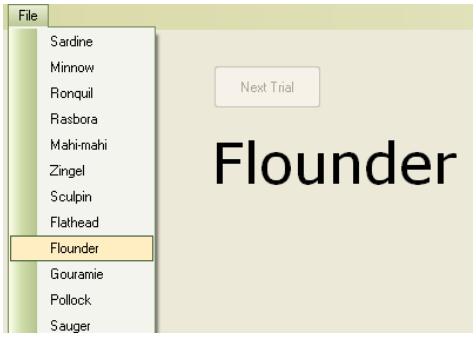

Figure 5. Experiment prototype with desktop menu.

with $Freq = 1$.

- *NB*: Usage of functions near the target. Values: low (neighbouring items are used zero times), high (a neighbouring item is used 16 times).

As we expect *Freq* to play a key role in determining model strength, we use a denser set of test values (than prescribed by Zipf). Using four items from a menu with $K = 40$ items mimics real-world scenarios with complicated interfaces where only 10% of the interface is used. Four of $K = 20$ items reflects a denser 20% functionality usage.

6.2 Recall Session

Once training is completed, participants have developed mental models $\theta^1, .., \theta^K$ for each of the K items in the test application (where $K = 20, 40$). The recall session is designed to allow estimation of mental model strengths by asking participants to carry out a series of *recall* tasks. Given menu item k, we asked participants to recall its location on a new menu, which has the same interface settings except menu labels are replaced with *filler* labels. The purpose of these filler labels is to see how well participants remember the actual location of the menu items. Fillers are designed to preserve the length of the original label using a random sequence of consonants. For example, the corresponding filler for "Flounder" is "Xtnxctzr".

To create the set of target items for this recall session, we first took the four targets from training and added their near-neighbours—i.e., all menu items within a distance of three—to the set. For example, if one of the original four targets was in location l, then items at positions $l + 1, \cdots, l + 3$ and $l - 1, \cdots, l - 3$ were added to the recall set as well. This yields a total of 28 ($= 4 \times 7$) items. However, the condition with high frequency neighbours (i.e., $NB = high$) require the target with $Freq = 16$ be located close to the other targets. Therefore, these 28 items are not distinct. As a result, the recall task has between 10 to 25 unique target items across all the conditions.

Participants were asked to recall each distinct item three times, resulting in 30 to 75 recall tasks presented in random order. In each task, participants were asked to identify the location of the targets as accurately and as quickly as possible. For each unique target, the participant responded with three samples of his mental model of that target's location (one per repeated trial), three corresponding response times, and a self-reported confidence score (how well he thinks he knows the original location, on a Likert scale).

6.3 Experiment Results

We used the task completion times in the training session to identify outliers so that if the completion times did not generally decrease for high frequency items, we assumed the participant did not pay attention. With this criteria, we discarded data from 5 participants and kept 48 for analysis. Next, we present the results from the recall session.

6.3.1 Computing Mental Model Strength

Recall our definition of model strength in terms of entropy is $1 - H(\theta^k)/H_L^+$ for an interface with L locations (here, $L = Length$). To compute strength, we estimated the model distribution using the recalled locations. For each unique target, the three responses were fit to a discrete normal distribution (fit using the sample mean and standard deviation of the three responses). Strength was computed using this distribution. This provided us with a single strength estimate, an average response time, and a confidence score for each of the 10 to 25 unique targets.

Across all participant-target pairs, the median strength is 0.40. We also measured the correlation between strength, confidence, and response time. We found that strength and confidence are positively correlated ($r = 0.46, p < 0.01$) while response time is not significantly correlated with either strength nor confidence ($r = -0.03$ and $r = 0.04$ respectively). This suggests our definition of strength assesses the participants' beliefs of how well they think they know the function locations.

6.3.2 Relevant Independent Variables

Using factorial ANOVA analysis on all the condition variables, we found that NB and $Freq$ have a significant effect on strength ($p < 0.01$) and $Length$ only marginally significant ($p < 0.1$). Thus, the variables NB and $Freq$ provide sufficient context to explain the data, so our strength estimate function is best modeled as $f(Freq^k, NB^k)$.

6.3.3 Quantitative Relationships

As we expected frequency to have less effect as it increases, we chose to fit the data using a logarithmic function. We fit a separate function for each value of NB. Fig. 6 shows the averaged data and the regression results. As presented in Sec. 4.2.1, we have $0.11 \log(0.51 Freq + 1) + 0.56$ with $r^2 = 0.9$ when $NB = high$, and $0.11 \log(0.44 Freq + 1) + 0.50$ with $r^2 = 0.9$ when $NB = low$. This is the function used to define the model strength induced with the new usage context in Sec. 3.2.

7. LEARNING THE COST OF DISRUPTION

The experiment described in this section attempts to assess the degree of disruption induced by adaptive actions in menu selection. It was conducted as a continuation of the Recall experiment in Sec. 6 with the same 48 participants.

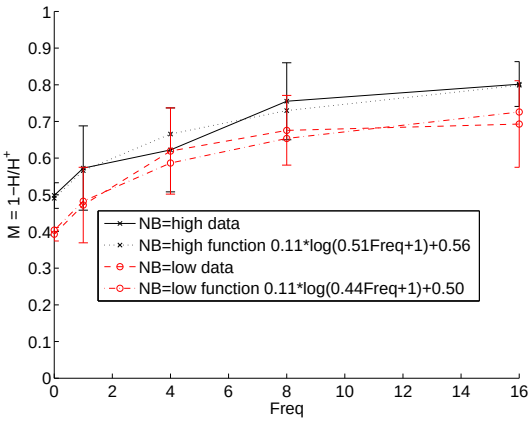

Figure 6. Resulting functions for mental model strength.

7.1 Experimental Set-Up

As defined in Sec. 4.2, the disruption of moving function k is given by $D_t^k(A)$. Our aim is to learn this function in the menu selection task, while restricting the action A to changing only one menu item at a time (i.e., $N = 1$). For convenience in this initial study, we treat disruption as additive in what follows, with total disruption being the sum of the disruptions over all functions. (The accuracy of this assumption will need to be verified in follow-up experiments.)

We trained the participant's mental model as described in Sec. 6.1 while keeping track of all task completion times. To induce disruption on the mental model, we applied one of the four adaptive actions (TOP, SWAP, HIDE, NONE) and then asked the participant to select the (potentially) moved target. Thereafter, we asked the participant to indicate whether the target was moved. If no, a self-reported disruption score of 0 was recorded. If yes, we further asked the participant to report the disruptiveness of that adaptation on a Likert scale. Since one of our adaptive actions is to hide menu items, the menu always has 10% of its items hidden. Otherwise, the same interface with different text labels was used.

For simplicity, we use a subset of the usage frequencies from Sec. 6 to create our target items. We randomly chose three target items and assigned them frequencies 1, 4, 16 respectively, creating 21 selection tasks with three distinct targets. We then augmented this set of tasks with adaptive actions and additional selection tasks as follows: after selecting a target p times (where p is the item's associated usage frequency), the system moves its location (as dictated by the chosen action A) and asks the participant to select the (potentially) moved target. With three target items, this adds three new tasks, yielding a total of 24 menu selection tasks.

Since we are interested in learning the disruption for every combination of usage frequency and system action under each experiment condition, we designed *four* sets of target items and associated selection tasks according to the above procedure. In total, we created 12 distinct target items and 96 menu selection tasks. Ideally, a separate experiment would be run for each combination of the condition variables and system action. However, the resulting protocol is too large, and would either be overwhelming for participants in a within-subjects experiment design or logistically infeasible for a between-subjects design. As a compromise, our conditions here vary only in $Length$ and $Freq$, and aggregated the other variables into one experiment. This experiment thus takes just an initial step in assessing disruption time.

7.2 Results

To estimate disruption time, we subtracted task completion time of the corresponding condition from the training phase from the task time in this new disruption phase. This gives us a crude assessment of the additional search time induced by the adaptive action. Correlation between disruption time and self-reported disruption scores are positive and significant ($r = 0.40, p < 0.01$). On average, we found the mean disruption time is about $1.5s$ with a disruption score of 2 (which corresponds to noticing a small amount of disruption), and the mean disruption time is about $6s$ with a disruption score of 5 (highly disruptive). We used these times to compute δ in Eq. (5).

Since this experiment is conducted following the Recall experiment, we took the estimated strength values from the same participant's corresponding conditions and used them in fitting $D_t^k(A)$. The data was noisy in general, so we binned the strength estimates into three equally-sized buckets and analyzed the disruption times with respect to a weak, medium-strength, and strong mental model. The mean values for these bins are 0.26, 0.66, and 0.90 respectively.

In general, we expect disruption time to increase as strength increases. We used the empirical disruption times for $A =$NONE as the baseline. For simplicity, we chose to fit the data using linear regression. Fig. 7 shows the averaged data and the following regression results: when $A = Swap$, we have $D = 1423M + 158$ with $r^2 = 0.5$, when $A = Top$, we have $D = 2865M$ with $r^2 = 0.8$, and when $A = Hide$, we have $D = 6033M + 2901$ with $r^2 = 0.9$. Overall, we see that

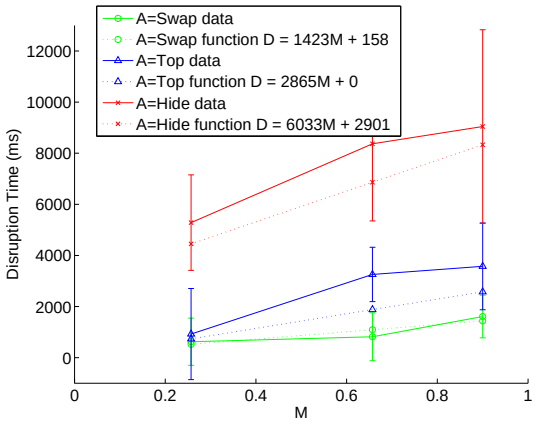

Figure 7. Resulting functions for disruption time.

disruption time is positively correlated with model strength, and most significantly so with HIDE. Note that existing predictive pointing models (e.g., Fitts law) do not account for a user's mental model, and at best attempt to reflect only a user's expertise level (e.g., [3]), neither of which adequately accounts for added disruption time. Our results suggest the need for such a model in adaptive systems.

8. USABILITY EXPERIMENT

We concluded this study with a usability experiment designed to test and verify the simulation results from Sec. 5 with real users, adopting model parameters estimated via the preceding experiments. We adopt the same set-up and evaluation metrics as those in the simulation experiments from Sec. 5. In total, we collected data from 8 participants.

8.1 Experimental Set-Up

Following the experiment in Sec. 5, we created menu selection tasks with these systems: BEST STATIC, RANDOM-4, SPLIT-4, and WER(w_s)-4. We chose BEST STATIC and RANDOM-4 as they provide baseline results, and SPLIT-4 as a plausible competitor to our model. Another competing approach not investigated here is the copying approach of Gajos et al. [6]—menu items are copied rather than moved to more convenient locations—which has been shown to be preferred by users. To offer a fair comparison, we could augment our adaptive WER(w_s)-4 policy with a COPY action. We leave this possibility to future research. To create the same visual menus in each system, the line typically separating the top and bottom partitions in split menus is removed. All parameter values used in this study are identical to those in Sec. 5.

There are two parts to this experiment. The first is a within-subjects experiment which asked participants to compare the 4 systems by carrying out 50 menu selection tasks with targets sampled from a Zipf frequency distribution. To help differentiate the experience, each system was designed with a different set of menu labels (e.g., fish, colors, fruits, animals). The second part follows the same design except it uses a uniform distribution rather than Zipf. In all cases, we used an interface similar to the one shown in Fig. 5 and fixed menu length to 20. The presentation order of the 4 systems and the two parts were counter-balanced across participants.

We let participants explore the interface using RANDOM-4 until they were comfortable. To determine their preference toward adaptive systems, we asked a multiple choice question, "Would you use adaptive menus if they were designed to SPEED UP the tasks?" To a response of "yes", we assigned the weight setting $w_s = .9$ in our WER system (with $w_d = 1 - w_s$), denoting the participant has a strong preference to maximize savings at the expense of added disruption. On the other hand, a response of "no" was assigned $w_s = .1$, denoting the participant has a strong preference to minimize disruption. Finally, a response of "maybe" was assigned $w_s = .5$. At the end of each part of the experiment, we asked participants to rate each system on a Likert scale based on frustration, ease of use, and efficiency.

8.2 Results

In each trial, we logged the task completion time (as opposed to the predicted selection time in the simulation evaluation) and estimated the corresponding disruption time. Following the format from the simulation, we report the objective usability results from the Zipf condition in Table 3. Among the 8 participants, 3 used the weight setting of $w_s = .9$ for our WER system, 4 used $w_s = .5$, and 1 used $w_s = .1$. Since we do not have an equal number of participants for each weight setting, we also aggregated their results together to provide an overall performance on WER.

Method	N	Task Time	Estimated Disrupt. Time	Total Strong Models	Percent Strong Moves
BEST STATIC	0	1513	0	134	0.0
RANDOM	4	2966	779	82	59.9
SPLIT	4	1760	26	111	9.8
WER(*all*)	4	1817	21	121	5.2
WER(.1)	4	2123	24	135	3.7
WER(.5)	4	1864	23	118	5.1
WER(.9)	4	1651	17	119	5.8

Table 3. Usability results using a Zipf distribution. Times in msec.

Method	N	Task Time	Estimated Disrupt. Time	Total Strong Models	Percent Strong Moves
BEST STATIC	0	2335	0	82	0.0
RANDOM	4	3322	993	54	50.9
SPLIT	4	3311	75	56	25.6
WER(*all*)	4	2913	47	60	29.7
WER(.1)	4	3546	53	84	1.2
WER(.5)	4	2792	27	63	33.5
WER(.9)	4	2865	31	49	34.2

Table 4. Usability results using a uniform distribution. Times in msec.

In general, we see similar results as those in the simulation: WER is competitive with SPLIT-4 when comparing task and disruption times. In contrast to the predicted selection times from the simulation results, the *measured* task times for the three adaptive systems are much higher. We suspect this effect is due to the participant's subjective annoyance factor toward the system's adaptations which resulted in an increased overhead. Unlike the simulation, WER(.9)-4, whose goal is to maximize savings, does better than SPLIT-4 on all dimensions. The t-test results show WER(.1)-4, whose goal is to minimize disruption, offers significantly more opportunities for learning strong models than SPLIT-4 ($p < 0.05$).

Significant advantages of our method are made more obvious when the tasks are created from a uniform distribution. Table 4 shows these usability results. Using t-test analysis, we see that WER(.5)-4 is significantly faster in task time than SPLIT-4 ($p < 0.05$) and WER(.1)-4 offers significantly more opportunities to develop strong mental models ($p < 0.01$).

Lastly, we report the post-questionnaire results in Fig. 8. Although no significance was found, in large part due to having to divide the number of participants into three WER weight cases, we see that on average, participants reported that our WER-4 system is less frustrating, easier to use, and more efficient than SPLIT-4. Overall, our usability results confirm and amplify the conclusions in the simulation experiment.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a probabilistic model of the user's mental model of function locations and defined three mental operations based on model strength. We implemented a decision-theoretic system that trades off the long-term savings of its adaptive actions with the costs of disruption, defined as a function of model strength. To model individual

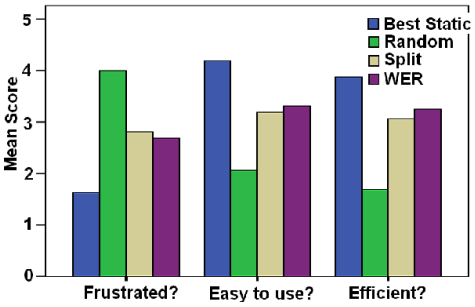

Figure 8. Subjective results

preferences, we parameterized our system with weights that capture a user's tradeoff between maximizing savings versus minimizing disruption. In addition, we conducted two empirical experiments to learn model parameters and evaluated the resulting model in simulation and with a usability study. Overall, our user adaptive approach respects user preferences, minimizes disruption of strong mental models, and is competitive with split menus in task selection performance in both simulation and the usability study.

A natural extension of this work is removal of the independence assumption over mental model distributions. Our results demonstrate value in estimating the cost of disruption as a function of model strength; we would like to further our approach by estimating disruption directly as a function of the mental model distribution itself. Due to the number of parameters involved, such an approach demands much more data than is empirically feasible (even with the simpler representation here we had to place constraints on our experiments.) Therefore, a tradeoff must be made between a more accurate, theoretical model and the ability to provide supporting, empirical evidence for it. Finally, investigation of richer disruption-sensitive adaptive policies (e.g., involving a wider space of actions, including a COPY action) would be of value, as would comparison to additional customization techniques (e.g., [6]).

Acknowledgement: Thanks to the anonymous referees for valuable suggestions. This work was supported by NSERC.

REFERENCES

1. J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. *IJCAI-05*, pp.1293–1299, Edinburgh, 2005.
2. J. Carroll and J. Olson. *Mental Models in Human-Computer Interaction*. National Academic Press, 1987.
3. A. Cockburn, C. Gutwin, and S. Greenberg. A predictive model of menu performance. *CHI-07*, pp.627–636, 2007.
4. H. Ebbinghaus. Memory: A Contribution to Experimental Psychology, 1885. Translated by H.A. Ruger and C.E. Bussenius. Retrieved on 01/08/2008 at http://psychclassics.yorku.ca/Ebbinghaus/.
5. P. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psych.*, 47:381–391, 1954.
6. K. Gajos, D. Christianson, R. Hoffmann, T. Shaked, K. Henning, J. Long, and D. Weld. Fast and robust interface generation for ubiquitous applications. *UBICOMP-05*, pp.37–55, Tokyo, 2005.
7. K. Gajos and D. Weld. SUPPLE: automatically generating user interfaces. *IUI-04*, pp.93–100, Madeira, Portugal, 2004.
8. D. Gentner and A. Stevens, eds. *Mental Models*. Lawrence Erlbaum, 1983.
9. S. Greenberg and I. Witten. Directing the user interface: how people use command-based computer systems. *IFAC Conference on Man-Machine Systems*, pp.349–356, 1988.
10. A. Hornof and D. Kieras. Cognitive modeling reveals menu search is both random and systematic. *CHI-97*, pp.107–114, 1997.
11. B. Hui and C. Boutilier. Who's asking for help? A Bayesian approach to intelligent assistance. *IUI-06*, pp.186–193, 2006.
12. B. Hui, S. Gustafson, P. Irani, and C. Boutilier. The need for an interaction cost model. *AVI-08*, pp.458–461, 2008.
13. P. Johnson-Laird. *Mental Models: Toward a Cognitive Science of Language, Inference and Consciousness*. Harvard Univ. Press, 1983.
14. D. Norman. Some observations on mental models. In D. Gentner, A. Stevens, eds., *Mental Models*. Lawrence Erlbaum, 1983.
15. L. Ross, M. Lepper, and M. Hubbard. Perserverance in self-perception and social perception: Biased attributional processes in the debriefing paradigm. *J. Personality and Soc. Psych.*, 32:880–892, 1975.
16. A. Rutherford and J. WIlson. Searching for mental models in human-machine systems. In *Models in the Mind*, pp.195–224. Academic Press, 1992.
17. M. Sasse. *Eliciting and Describing User's Models of Computer Systems*. PhD thesis, School of Computer Science, University of Birmingham, Birmingham, England, 1997.
18. A. Sears and B. Schneiderman. Split menus: Effectively using selection frequency to organize menus. *ACM Trans. on Computer-Human Interaction*, 1(1):27–51, 1994.
19. G. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, 1949.