

When Actions Have Consequences: Empirically Based Decision Making for Intelligent User Interfaces

Anthony Jameson Barbara Großmann-Hutter Leonie March Ralf Rummer
Thorsten Bohnenberger Frank Wittig


Department of Computer Science / Department of Psychology, Saarland University
P.O. Box 15 11 50, 66041 Saarbrücken, Germany

jameson@cs.uni-sb.de barbara@cs.uni-sb.de r.rummer@rz.uni-sb.de

bohlenberger@cs.uni-sb.de fwittig@cs.uni-sb.de

Fax: +49 681 302-4136, Phone: +49 681 302-2363 / 302-3196

ABSTRACT

One feature of intelligent user interfaces is an ability to make decisions that take into account a variety of factors, some of which may depend on the current situation. This article focuses on one general approach to such decision making: Predict the consequences of possible system actions on the basis of prior empirical learning, and evaluate the possible actions, taking into account situation-dependent priorities and the tradeoffs between the consequences. This decision-theoretic approach is illustrated in detail with reference to an example decision problem, for which models for decision making were learned from experimental data. It is shown  influence diagrams and methods of decision-theoretic planning can be applied to arrive at empirically well-founded decisions. This paradigm is then compared with two other paradigms that are often employed in intelligent user interfaces. Finally, various possible ways of learning (or otherwise deriving) suitable decision-theoretic models are discussed.

Keywords

Intelligent user interfaces, decision theory, Bayesian networks, machine learning

1 INTRODUCTION

One of the properties that distinguish intelligent user interfaces (IUIs) from more mainstream user interfaces is their ability to make decisions that go beyond what the user has instructed them to do: They take into account a relatively large number of factors, and they arrive at their decisions by methods that cannot be described in terms of straightforward algorithms—typically making use of some explicit representation of the goals and properties of the user and the current context (cf. [23]).

The decisions that IUIs can face vary widely. They can concern, for example:

1. the generation of a complex presentation, such as a graph-

ical presentation of data that will achieve a given communicative goal ([18]);

2. the recommendation of one or more appropriate objects (e.g., web pages or products) from a very large set of objects ([21]; [4]);
3. a choice among a small number of possible ways to achieve a given result (e.g., whether to use checkboxes or radio buttons to elicit Boolean data from the user: [9]);
4. a yes-or-no decision as to whether to perform a given optional action or not (e.g., whether or not to pass a given message on to a user who may not be in a good position to process it: [12]).

This article focuses a specific approach to decision making, the one characterized in Table 1. When contrasting this method with other approaches to decision making for IUIs, we will refer to this approach simply as the *decision-theoretic approach*—although the term *decision-theoretic* is used in different ways in various scientific fields. As we will see, this approach is most applicable to cases like the third and fourth ones in the list above. In particular, each decision tends to be fine-grained and to represent a small part of the system's overall processing. Still, the overall impact of many such fine-grained decisions can be important.

A number of publications are available that include descriptions of techniques such as the ones discussed here in the context of descriptions of entire systems (see, e.g., [13]; [11]; [16]). In the present article, to focus attention on questions concerning the methods themselves, we start by describing a simple example system, a type of decision that this system has to make regularly, and an experiment in which we gathered data relevant to this type of decision. Section 3 shows how single decisions can be made in this limited domain with decision-theoretic tools, and Section 4 shows how more complex sequences of interrelated decisions can be handled. Section 5 compares this paradigm with two other approaches, and Sections 6 and 7 consider possible variants of the paradigm.

Single decisions

Situation

- There is a relatively small number of possible actions for the system \mathcal{S} .
- It is possible for \mathcal{S} to take into account the possible *consequences* (including the costs) of these actions.
- Which particular consequences the action will have in a given case depends on various factors, and the consequences cannot all be predicted with certainty.
- There are evaluative tradeoffs among the consequences of an action.
- The relative importance of the different consequences for the user \mathcal{U} varies from one situation to the next.

Method

- Develop a causal model for predicting the consequences of actions, if possible on the basis of empirical data.
- Choose an action by using decision-theoretic methods which make use of the causal model.

Sequences of decisions

Situation

[Same conditions as for single decisions, plus the following conditions:]

- A sequence of related individual decisions has to be made.
- The desirability of each of the actions chosen may depend on the consequences of previous actions.

Method

- Frame the problem in terms of Markov decision processes, again using empirical data if possible.
- Use methods from decision-theoretic planning to arrive at plans or policies, depending on whether feedback about the results of individual actions will be available.

Table 1. Overview of the decision-theoretic approach to decision making examined in this article.

2 EXAMPLE DOMAIN AND EXPERIMENT

2.1 The Specific Adaptation Issue

The usage scenario addressed by our example system is illustrated in Figure 1: The user \mathcal{U} is to select appropriate options from part of a printing dialog box; but since she doesn't know the options, the system \mathcal{S} gives spoken instructions that are appropriate for the current situation. The complex processing of such a system concerns the problem of determining which particular instructions are appropriate. The small decisions that we will look at concern the way in which these instructions are presented: Should several be presented at a time, or should each instruction be presented individually?

To perform a controlled empirical study of this question, we first devised the more abstract, artificial dialog box shown in Figure 2. In this dialog box, factors such as the users' prior knowledge of the meaning of the options are kept out of the picture—a fact that will make the analyses in the following sections easier to understand.¹ In the artificial system, each instruction has the form “Set <letter> to <number>.” Two

¹The problem of generalizing the results to real systems will be discussed in Section 6.2.

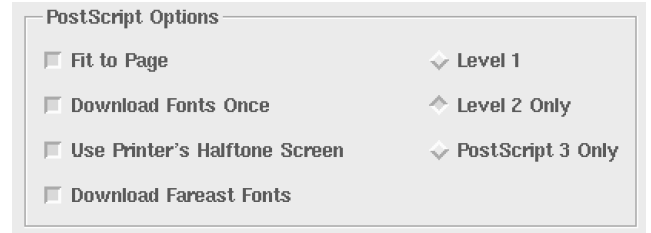


Figure 1. Typical dialog box (from the Adobe Acrobat Reader) for which a user might require a sequence of instructions.

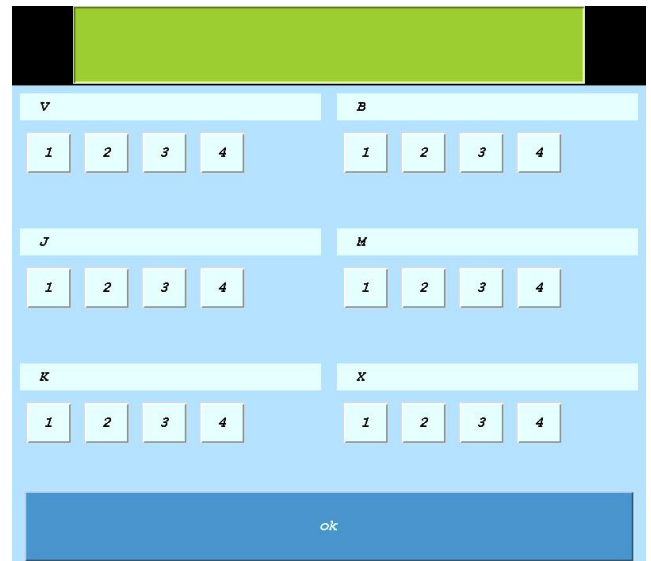


Figure 2. Main screen used for the experiment.

possible ways of presenting a set of three instructions are shown in Figure 3: (a) in a *stepwise* mode (i.e., allowing \mathcal{U} to execute each instruction in the sequence before hearing the next one) or (b) in a *bundled* mode (i.e., all at once, before \mathcal{U} starts executing the first instruction).

The main drawback of stepwise presentation is the additional interaction overhead: After executing each instruction, \mathcal{U} must somehow confirm to \mathcal{S} that he is ready for the next instruction—here, by clicking on the “OK” button. (We are assuming here that \mathcal{S} does not get direct information about \mathcal{U} 's task performance.) This confirmation signaling requires

<i>Stepwise:</i>	<i>Bundled:</i>
Set X to 3	\mathcal{S} : Set X to 3
... [OK]	\mathcal{S} : Set M to 1
Set M to 1	\mathcal{S} : Set V to 4
... [OK]	
Set V to 4	

Figure 3. Illustration of the two presentation modes for instructions.

a certain amount of time and effort on \mathcal{U} 's part.²

The main limitation of bundled presentation is that it may require \mathcal{U} to try to store an excessive amount of information in working memory (WM). If \mathcal{U} 's available WM capacity is inadequate—for example, because the sequence of instructions is especially long, or because \mathcal{U} simultaneously has to store unrelated information in WM— \mathcal{U} may fail to remember the instructions. The resulting errors in task performance may far outweigh the time saved by bundling the instructions. Accordingly, we may expect bundled presentation to be inappropriate if the sequence of instructions is especially long, and/or if \mathcal{U} 's effective WM capacity is temporarily limited because \mathcal{U} is distracted by environmental stimuli and/or a task that he has to perform simultaneously.³

If we were developing a full-blown, complex system, we wouldn't want to dwell much longer on this one aspect of its behavior. It would be consistent with current practice to implement a decision rule more or less like the following one:

- If \mathcal{U} is significantly distracted, or if the sequence of instructions comprises more than 3 steps, then use stepwise presentation;
otherwise, use bundled presentation.

In an effort to develop an especially well-founded decision procedure for this example system, we took the time to collect extensive empirical data in a controlled setting.

2.2 Method

2.2.1 Materials

Figure 2 shows the screen that subjects worked with throughout the experiment. They used a mouse to click on the buttons labeled with digits and on the large *OK* button.

Their primary task was to execute sequences of spoken instructions, each sequence comprising 2, 3, or 4 *steps*. Each sequence was presented by the system in either *stepwise* or *bundled* mode (see Figure 3).⁴ In stepwise mode, after executing a single instruction, the subject had to signal completion by clicking on the *OK* button in order to receive the instruction for the next step. Each individual instruction for a step was played from a separate sound file; the sound files played for a given sequence in stepwise and bundled modes were identical, the only difference between the modes being the ordering of the actions of subject and system.

On half of the trials, the large rectangle at the top of the

screen provided a situational distraction which was intended to reduce the amount of working memory capacity that the subject had available for the primary task: At more or less regular intervals, the rectangle took on a color that alternated between red and green in random order. Whenever the same color appeared twice in succession, the subject was to press the space bar.

2.2.2 Design

There were three independent variables:

- PRESENTATION MODE: stepwise or bundled
- DISTRACTION?: no or yes
- NUMBER OF STEPS: 2, 3, or 4

Twelve specific experimental conditions were created through orthogonal combination of these factors.

Two dependent variables will be discussed here:⁵

- EXECUTION TIME

In both conditions the execution time was the total time required for the processing of an instruction sequence, minus the time required by the system to play the instructions.⁶

Specifically: In bundled mode, this was the duration of the interval between (a) the moment the system finished playing the instruction sequence and (b) the moment the subject completed the final step in the sequence by pressing one of the numbered buttons. In stepwise mode, it was the sum of the corresponding durations for the individual steps, including the time required to press the *OK* button after performing each instruction except the last one.

- ERROR?

This variable had the value 0 for a particular instruction sequence if the subject pressed all of the instructed buttons in the correct order, the value 1 otherwise.⁷

2.2.3 Subjects

Subjects were 24 students from various departments at Saarland University, who received financial compensation for their time.

2.2.4 Procedure

For each subject, the experiment began with a practice phase that was intended to familiarize subjects with the experimental environment and minimize learning during the main part of the experiment. Four blocks of instruction sequences were introduced and practiced in turn, each involving one combination of the independent variables PRESENTATION MODE and DISTRACTION?.

In the main phase of the experiment, the instruction sequences were again presented in four blocks, whose order

²Moreover, \mathcal{S} may be able to formulate the instructions more concisely if \mathcal{S} can bundle them together, for example by using ellipsis.

³Note that this discussion is not limited to systems that use speech to present instructions. Similar considerations are relevant with other ways of presenting instructions, such as animations or videos, as long as it is \mathcal{S} that controls when \mathcal{U} can perceive the instructions (i.e., \mathcal{U} cannot switch back and forth at will between task performance and perception of the instructions, as he might using a conventional help system that presents static text and images).

⁴The original German formulation of an instruction was "Setze <letter> auf <number>."

⁵Further dependent variables are analyzed in [22].

⁶Note that it would just as well have been possible to include the total time for the playing of the instructions, since this time was identical in stepwise and bundled mode.

⁷This definition is appropriate for domains in which performing one action incorrectly is just as bad as performing all actions incorrectly. In many domains, of course, the number of errors would be a more important variable.

was systematically varied across subjects. In each block, 18 instruction sequences were presented, in 3 subblocks, each of which comprised 6 sequences of each length. Half of the subjects started with the shorter sequences and moved on to the longer ones, while for the other half the order was reversed. Thus in all, data on 72 sequences were obtained from each subject, with each specific condition being represented by 6 sequences.

2.3 Results

Most interesting for our purposes are the results for the 12 specific combinations of independent variables, shown in Figure 4. But to give an idea of the statistical reliability of the results, we also report on statistical analyses of the effects of individual independent variables.⁸

First, a multivariate analysis of variance was conducted with NUMBER OF STEPS, PRESENTATION MODE, and DISTRACTION? as within-subject independent variables and EXECUTION TIME and ERROR? as dependent variables. All three main effects were significant.⁹ All interactions were also significant¹⁰. It is therefore appropriate to interpret the univariate analyses of variance reported below.

2.3.1 Execution Time

With respect to execution time, main effects were found for all three independent variables: Execution time is on the whole longer if there are more steps in the sequence ($F(2, 46) = 429.568, p < 0.001$); if the presentation is stepwise ($F(1, 23) = 51.141, p < 0.001$); and if there is a distraction task ($F(23, 1) = 38.705, p < 0.001$).

The increase with the length of the instruction sequence is easily understandable in view of the larger number of actions that need to be performed.

The difference between the two presentation modes is due mainly to the additional interaction overhead associated with stepwise presentation. This overhead consists in 1, 2, or 3 extra clicks on the OK button, respectively, for sequences of length 2, 3, and 4. Since this extra overhead is greater for longer sequences, it is understandable that there is a significant two-way interaction between NUMBER OF STEPS and PRESENTATION MODE ($F(2, 46) = 23.727, p < 0.001$).

The longer execution times when there is a distraction task are explainable at least in part simply in terms of the greater number of keypresses required in this condition.

There is a significant interaction between NUMBER OF STEPS and DISTRACTION? ($F(2, 46) = 6.687, p < 0.003$): DISTRACTION? increases EXECUTION TIME more when the sequence is long, in part simply because \mathcal{U} has to respond to the sec-

⁸Readers who are not interested in the conventional analysis of the data can simply look at Figure 4 and skip to the end of Section 2.

⁹ $F(4, 20) = 122.582, F(2, 22) = 95.192,$ and $F(2, 22) = 19.785$ respectively, with $p < 0.001$ in all three cases.

¹⁰NUMBER OF STEPS \times PRESENTATION MODE: $F(4, 20) = 39.713, p < 0.001$; NUMBER OF STEPS \times DISTRACTION?: $F(4, 20) = 3.612, p < 0.05$; PRESENTATION MODE \times DISTRACTION?: $F(2, 22) = 12.545, p < 0.001$; three-way interaction: $F(4, 24) = 3.842, p < 0.05$.

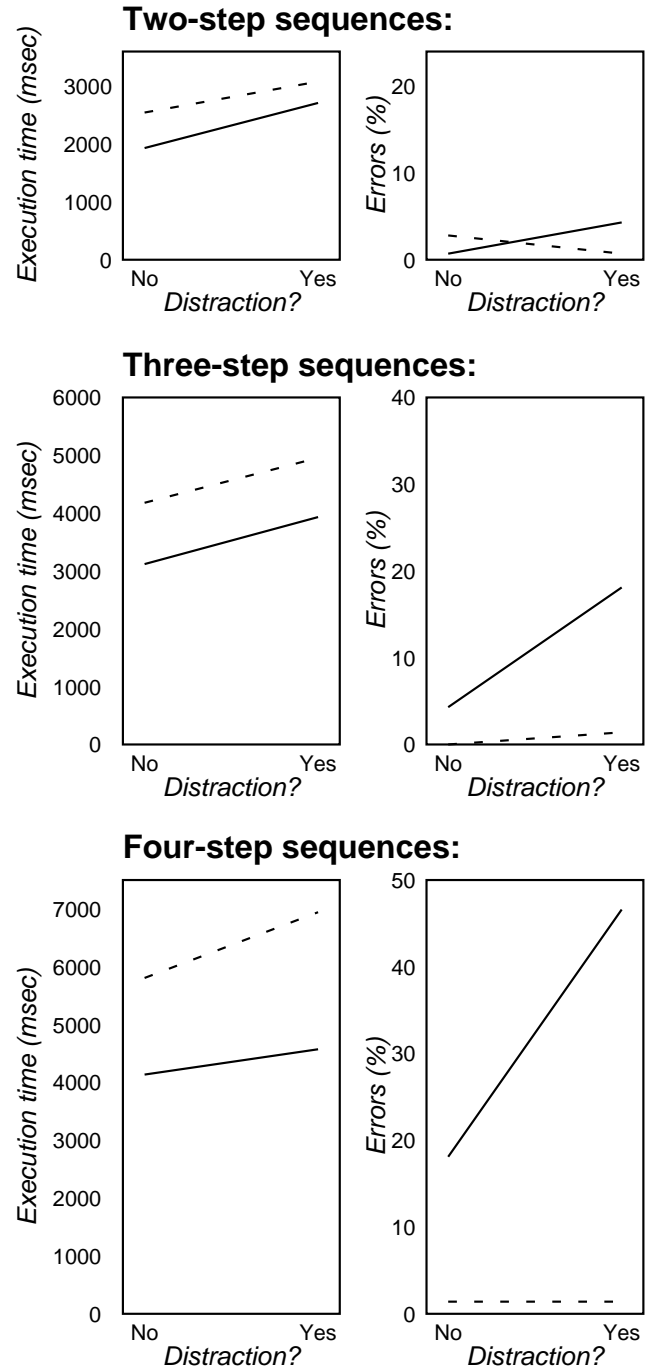


Figure 4. Mean execution times and error rates for each combination of values of the independent variables.

(Dashed lines represent stepwise presentation, solid lines bundled presentation.)

ondary task a larger number of times.

There are no other significant interactions involving execution time.

In sum, the results for the variable EXECUTION TIME can be understood fairly straightforwardly in terms of the number of physical actions that \mathcal{U} has to execute in the various conditions.

2.3.2 Errors

With respect to errors, main effects were again found for all three independent variables: The probability of an error is greater if there are more steps in the sequence ($F(2, 26) = 51.009, p < 0.001$); if the presentation is bundled ($F(1, 23) = 54.064, p < 0.001$); and if there is a distraction task ($F(1, 23) = 17.214, p < 0.001$).

In addition, all of the possible statistical interactions involving ERROR? were significant¹¹. Simply put, when two or more conditions co-occur that tend to produce errors, the number of errors becomes greater than the sum of the numbers that would be produced by these conditions individually. The extreme case is the 46.6% probability that \mathcal{U} will make at least one error in the most unfavorable condition (see the bottom right-hand graph in Figure 4).

2.4 Brief Discussion

In sum, a conventional analysis of the data confirms the qualitative hypotheses formulated at the beginning of this section: Stepwise presentation of instructions, unlike bundled presentation, is a slow but safe method which is essentially invulnerable to situational distractions.

3 USING THE RESULTS IN A LEARNED INFLUENCE DIAGRAM

The experiment just reported on tells us more than we would normally have time to find out about the causal relationships that are relevant to \mathcal{S} 's decisions about how to present instructions. But we still don't have a decision procedure that specifies exactly when \mathcal{S} should present its instructions in a stepwise (vs. bundled) mode. We will now present and motivate a way of deriving such a decision procedure.

3.1 Learning a Bayesian Network

The first thing we need is a model that will help \mathcal{S} to predict \mathcal{U} 's execution time and errors in each specific situation. For this purpose, we defined a Bayesian network (BN, [25]) with the structure shown in Figure 5. Each node in this network is observable: For each observation in the dataset resulting from the experiment (describing how a particular subject handled a particular sequence of instructions), a precise value of the corresponding variable is available.

Learning a BN that contains only observable variables and

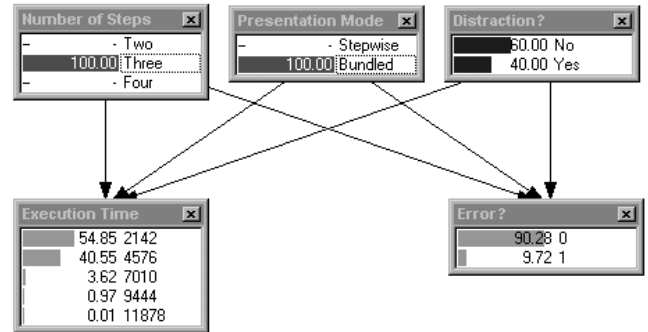


Figure 5. Bayesian network learned on the basis of the experimental data, showing a prediction made under uncertainty about the independent variable DISTRACTION?.

(The histogram for each variable shows a probability distribution that represents the system's "belief" about the value of that variable. Links represent causal influences. The numbers in the right-hand column of the window for EXECUTION TIME are the midpoints (in msec) of the five intervals defined for this variable.)

whose structure has been specified in advance is straightforward. In particular, the maximum likelihood estimate for each (conditional) probability can be computed simply in term of the (relative) frequencies in the data (see, e.g., [6]).¹² Figure 5 shows a screen shot of the learned BN as it is implemented in the decision-theoretic tool HUGIN (<http://www.hugin.com>)

The properties of the data that are summarized in Figure 4 are reflected in the conditional probability tables (CPTs) of the learned BN and in the specific inferences made by the BN. In particular, the basic uncertainty-management capabilities of BNs allow the following generally useful types of inference:

1. *Predicting the dependent variables given uncertainty about the values of the independent variables:* \mathcal{S} may want to make a prediction (and perhaps a related decision) without knowing the values of all of the independent variables shown in Figure 5. For example, \mathcal{S} may not know whether \mathcal{U} is currently distracted. As Figure 5 shows, if a probability distribution for each of these variables is specified, the network will still generate predictions for \mathcal{U} 's execution times and errors.

2. *Learning about \mathcal{U} or the current situation:* When \mathcal{U} 's performance in performing a task is observed, the corresponding node(s) (EXECUTION TIME and/or ERROR?) can be instantiated with the observed values. Evaluation of the net will then lead to updated beliefs about any variables that were not already known with certainty. For example, Figure 6 shows

¹¹NUMBER OF STEPS \times PRESENTATION MODE: $F(2, 46) = 41.708, p < 0.001$; NUMBER OF STEPS \times DISTRACTION?: $F(2, 46) = 7.877, p < 0.001$; PRESENTATION MODE \times DISTRACTION?: $F(1, 23) = 16.828, p < 0.001$; three-way interaction: $F(2, 46) = 8.793, p < 0.01$.

¹²We have also used these same data to learn considerably more complex Bayesian networks, whose additional nodes represent (a) other variables measured in the experiment, such as errors on the distraction task; (b) a variable representing the average execution speed of the current subject, which makes it possible to take individual differences into account; and (c) an unobservable node representing \mathcal{U} 's current working memory load (see [33]; [10]). To focus attention on the central methodological issues, we discuss here only the simplest possible network that could be defined for this experiment.

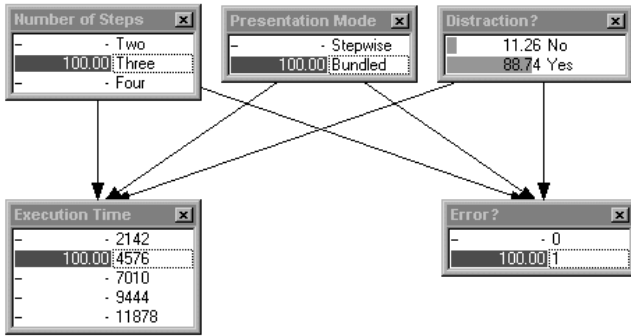


Figure 6. The same network as in Figure 5, showing the interpretation of an observation of U 's performance.

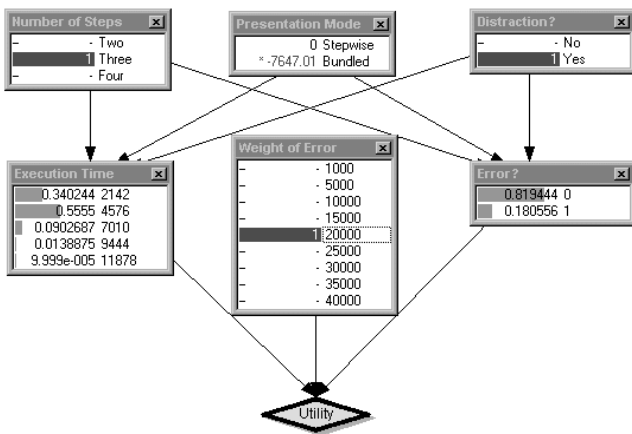


Figure 7. An influence diagram defined as an extension to the BN of the previous figures.

(The variable WEIGHT OF ERROR represents the amount of additional execution time, in msec, that it would be worth accepting in order to avoid an error; the numbers in the right-hand column of the window for this variable are the midpoints of the five intervals defined for it.)

how the network infers that S was probably distracted after observing that U made an error. This learning about the current U can enhance the quality of S 's future predictions and decisions about this particular U .¹³ In terms of the original experiment, this type of inference is like choosing a random observation about a subject and trying to guess what specific condition the subject was in when he or she generated that observation—a type of inference which is not supported by the usual techniques for analyzing experimental data.

3.2 Extending the BN to an Influence Diagram

Before the system can actually make decisions, we have to extend this network to an influence diagram (Figure 7), in two steps:

1. Add a *value* (or *utility*) node that expresses the system's evaluation of a particular combination of values of the de-

¹³If this type of learning about the user is to be done repeatedly, the BN has to be extended to become a dynamic BN—see, e.g., [16].

pendent variables ERROR? and EXECUTION TIME. Note that the relative importance of these two criteria can vary greatly depending on the nature of the task (e.g., setting font preferences vs. controlling a power plant) and the situation. If this relative importance were fixed once and for all, it could be encoded directly into the CPT of the utility node. Instead, to make it possible to use different importance weights, we introduce a variable WEIGHT OF ERROR. S 's (possibly uncertain) belief about this variable reflects S 's assumptions about the current priorities. For example, a value of 15,000 means that avoiding an error is just as important as saving 15,000 msec (i.e., 15 seconds) of execution time.

2. Make PRESENTATION MODE into a *decision node*. This change makes explicit the fact that S can freely choose the value of this variable, instead of just forming a belief about it.

Once the complete influence diagram has been specified, S can have it *evaluated* in order to determine the utility of each action it might take in each situation.¹⁴ For example, Figure 7 shows that the expected utility of the bundled presentation of three instructions given DISTRACTION? is -7647 . Apart from the minus sign, value is essentially the sum of (a) the expected execution time (in msec) and (b) a penalty that reflects the possibility that U will make an error. By comparing this utility with that of stepwise presentation, S can decide which mode to use.

In addition to having S make decisions in individual cases, the designer of an IUI will probably want to have an overall picture of the decisions that S will make. For example, it's conceivable that, for any reasonable value of WEIGHT OF ERROR, S would always decide to use stepwise presentation. In that case, the designer could probably save a lot of trouble by not implementing bundled presentation in the first place. Some tools for evaluating influence diagrams¹⁵ offer a way of getting such an overview without iterating through all possible situations and seeing what decision is recommended by the influence diagram: Associated with each decision node is a table that describes the decision rules¹⁶ for that node—i.e., what decision should be made for each possible combination of the values of the variables that may be known precisely at the time of the decision. The rules for the present influence diagram are summarized concisely in Table 2.

Note that the type of information provided by the rules for a decision node can have a function similar to that of the results of the sensitivity analyses that are sometimes performed with predictive models of interface designs. For example, [7, Chap. 7] includes a sensitivity analysis which determined in what situations a new method for correcting typing mistakes in an editor would be more effective than the already exist-

¹⁴For discussions of the relevant algorithms, see [30] and [17].

¹⁵For example, see NETICA, a commercial tool available from <http://www.norsys.com/>.

¹⁶The term *policy* is sometimes used to refer to these rules; but we will not use it here, since its meaning is different from that of the term *policy* that will be introduced in Section 4.

Steps	Without Distraction		With Distraction	
	Weights	Plan	Weights	Plan
2	≥ 1	(2)	1–10	(2)
			> 10	(1+1)
3	1–30	(3)	1–5	(3)
		> 30	(1+1+1)	> 5
4	1–5	(4)	1	(4)
		> 5	(1+1+1+1)	> 1

Table 2. Rules for deciding between bundled—e.g., (2)—and stepwise—e.g., (1+1)—presentation of an instruction sequence, for various combinations of parameters.

(The number(s) to the left of each presentation mode refer to the level(s) of importance of error-free performance for which that mode is chosen—in units of seconds, rather than milliseconds as in Figure 7. The notation employed here is used to facilitate comparison with the more complex Tables 3 and 4 below.)

ing methods. It is a convenient feature of some influence diagram tools that they produce information of this sort as a side effect in situations such as the one considered here.

In sum, the methods that we have applied so far are straightforward, in that they combine well-known methods from experimental psychology with some of the simpler functions of readily available decision-theoretic software tools. Yet they permit a more effective use of the experimental data for decision making than would be possible through the use of a conventional data analysis.

4 PLANNING SEQUENCES OF DECISIONS

Up to now, we have been assuming that \mathcal{S} was confronted with a binary decision: whether to present instructions in a bundled or in a stepwise manner. More generally, the methods discussed apply when a choice needs to be made from among a reasonably small number of mutually exclusive actions.

But our \mathcal{S} should actually have more flexibility in the example task considered: For example, to present 4 instructions, it should be able to present 2 bundles of 2 instructions each.

If we allow this type of solution, we are faced with a different type of decision problem, but one which likewise can occur in many types of IUI: We can frame the problem in terms of Markov decision processes (MDPs; see, e.g., [3]; [27], chap. 17).

Figure 8 characterizes the problem in such a way that it can be handled by an MDP. The problem is framed as one of how best to move \mathcal{U} through a space of states by performing actions. In our domain, each state is defined in terms of four features, each of which is described by one box in Figure 8.

Each action that \mathcal{S} can take leads to a transition to another state with a given probability. Figure 8 summarizes the principles that determine which states can be reached from a given state and what the transition probabilities are.

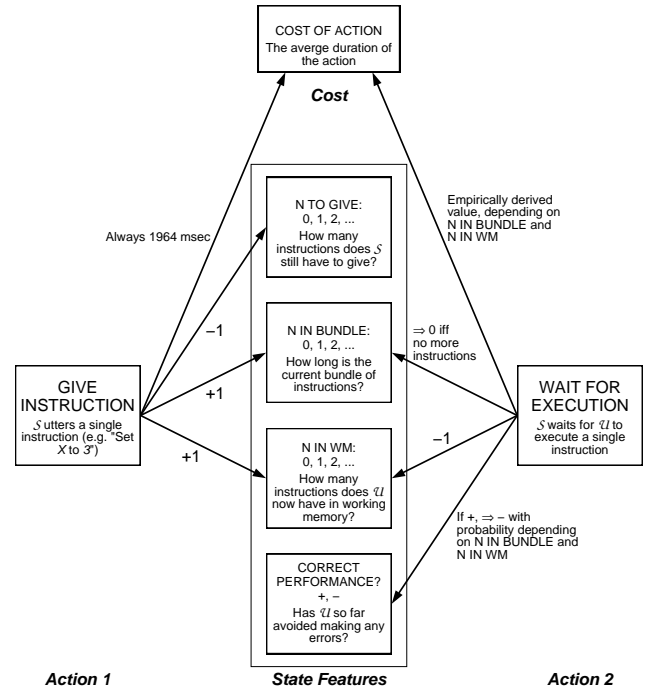


Figure 8. Summary of the modeling of the example problem as a Markov decision process.

(Each arrow shows a consequence of one of the two possible system actions in terms of either time cost (top) or a change in a feature of the current state.)

For the simple task of presenting only two instructions, Figure 9 shows the resulting state transition graph. (The graphs for larger numbers of instructions are similar, but they contain much larger numbers of states.) The two states on the right are the terminal states which are reached after all instructions have been executed. The upper one, corresponding to correct execution, is associated with a reward: a number that reflects how important error-free task completion is. This quantity is comparable to the variable WEIGHT OF ERROR that appeared in our influence diagram. For example, if its value is greater than 10, it is better for \mathcal{U} to take 10 seconds longer to complete the task than for her to make an error. (Without loss of generality in this case, we assume that the reward for completion with at least 1 error is 0.)

4.1 Plans for Presentations Without Feedback

We first consider the case where \mathcal{S} gives all of its instructions without receiving any feedback on whether \mathcal{U} executed them correctly or not. \mathcal{S} 's goal is to find an optimal plan for presenting the instructions, by considering what might happen when particular plans are used. For example, in Figure 9 we see that the plan of presenting the two instructions in a single bundle can result in three possible paths through the graph (the ones that pass through the uppermost state in the third column), depending on \mathcal{U} 's success in executing each instruction. Only the topmost path, which is by far the most likely one, ends in the state in which all instructions have been executed correctly.

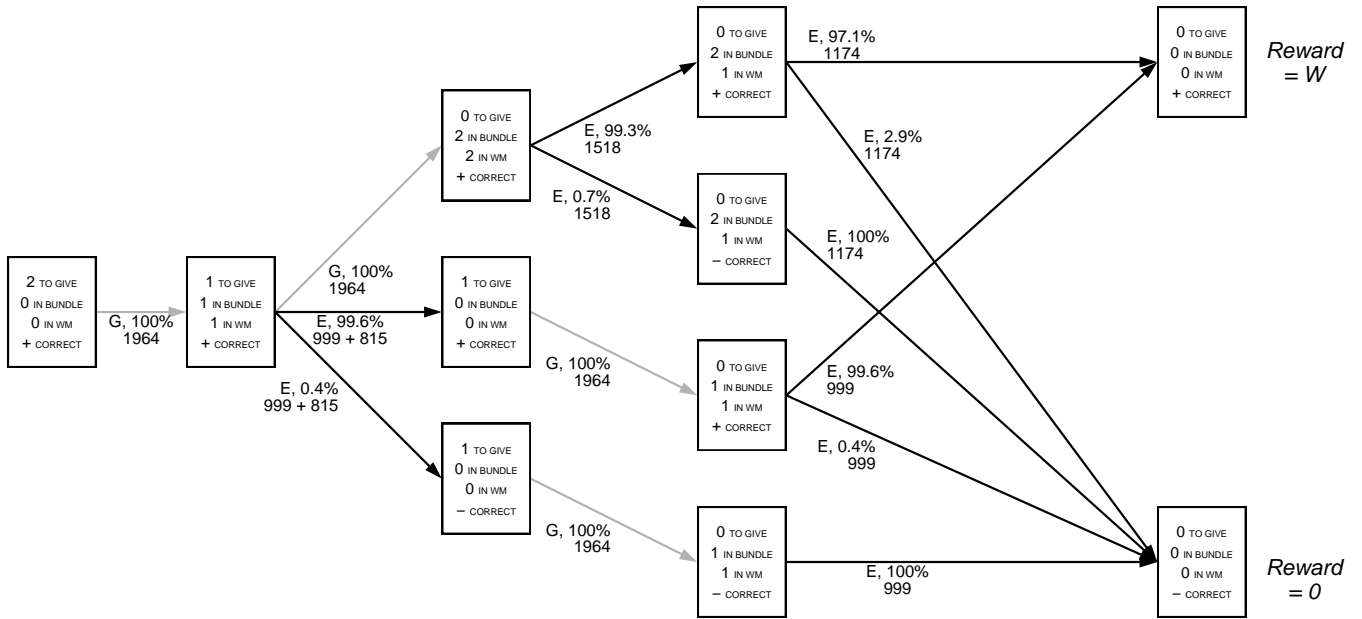


Figure 9. States and transitions for the case of two instructions.

(The label for each arrow indicates the action (“G, Give Instruction”, with gray arrows, or “E, Wait for Execution”, with black arrows); the probability that the action leads to the transition shown (always 100% for “G”), and the time cost of the transition (in msec). In this figure, these parameters are based on the data from the experimental condition with distraction by a secondary task. The features listed for the states are explained in Figure 8.)

A straightforward procedure for finding the optimal plan is the following one:

1. Construct all possible plans.
2. For each plan,
 - determine the set of possible paths through the graph that might result from the execution of this plan;
 - for each such path consider its likelihood and its net reward (the difference between its reward and its costs);
 - use this information to compute the expected utility of the plan.
3. Choose the plan with the highest expected net utility.

Table 3 shows the plans that are chosen given various combinations of parameters, including cases where the number of instructions is greater than the number that was included in any condition of our experiment. When the number of instructions is 2, 3, or 4, the choices are similar to those that were made with the simpler modeling, except that in some cases \mathcal{S} takes advantage of the possibility of breaking the sequence up into subsequences of 2 or 3 instructions.

As with the previous modeling, when DISTRACTION? is present \mathcal{S} tends to choose slower, safer plans.¹⁷

Steps	Without Distraction		With Distraction	
	Weights	Plan	Weights	Plan
2	≥ 0	(2)	0–4	(2)
			> 4	(1+1)
3	0–15	(3)	0–4	(3)
	> 15	(2+1)	> 4	(1+1+1)
4	0–1	(4)	0–2	(4)
	> 1	(2+2)	3–4	(3+1)
			> 4	(1+1+1+1)
6	0	(4+2)	0–1	(4+2)
	1–6	(3+3)	2–4	(3+3)
	> 6	(2+2+2)	> 4	(1+1+1+1+1+1)
8	0–1	(4+4)	0–3	(4+4)
	2–6	(3+3+2)	4	(3+3+2)
	> 6	(2+2+2)	> 4	(1+1+1+1+1+1+1+1)
10	0	(4+4+2)	0–2	(4+4+2)
	1	(4+3+3)	3–4	(3+3+3+1)
	2–6	(3+3+2+2)	> 4	(1+1+1+1+1+1+1+1+1+1)
	> 6	(2+2+2+2+2)		

Table 3. Plans for instruction presentation without feedback, for various combinations of parameters.

(“(L + M)” denotes a bundle of L instructions followed by another bundle of M. Each range in the column “weights” shows the level(s) of importance of error-free performance for which the corresponding plan is chosen.)

¹⁷A few apparent deviations from this pattern can be found in the table, but the plans shown are in fact optimal given the specific parameters that were derived from the empirical data. With a different sample of data, some parameter estimates would be different, especially where very low error probabilities are involved.

Steps	Without Distraction		With Distraction	
	Weights	Basis for Policy	Weights	Basis for Policy
2	≥ 0	(2)	≥ 0	(1)
3	0–8	(3)	≥ 0	(1+1+1)
	> 8	(1+2)		
4	0–2	(4)	0–2	(4)
	> 2	(2+2)	> 2	(1+1+1+1)
6	0–4	(4+2)	0–4	(4+1+1)
	> 4	(2+2+2)	> 4	(1+1+1+1+1+1)
8	0–2	(4+4)	0–2	(4+4)
	3–6	(4+2+2)	3–7	(4+1+1+1+1)
	> 6	(2+2+2+2)	> 7	(1+1+1+1+1+1+1+1)
10	0–4	(4+4+2)	0–4	(4+4+1+1)
	4–8	(4+2+2+2)	5–9	(4+1+1+1+1)
	> 8	(2+2+2+2+2)	> 9	(1+1+1+1+1+1+1+1+1+1)

Table 4. Policies for instruction presentation with feedback for various combinations of parameters.

(The notation is as in Table 3. The basis for the policy is the grouping of instructions that \mathcal{S} uses as long as no error occurs; as soon as negative feedback is received, \mathcal{S} presents the remaining instructions in bundles of maximal length.)

4.2 Policies for Presentation With Feedback

There are many scenarios in which our assistance system \mathcal{S} could receive immediate feedback on how \mathcal{U} had executed a given instruction. In such a case, it makes little sense for \mathcal{S} to derive and carry out a complete plan for presenting the instruction sequence; instead, it should be prepared to adapt its behavior to the feedback that it gets from \mathcal{U} . It still makes sense for \mathcal{S} to look ahead and anticipate what might happen; but the result of this process will be not a plan but a *policy*, which specifies what \mathcal{S} should do under various circumstances. (In the terminology of decision-theoretic planning, \mathcal{S} will always be aware of the current state, and the process is a *fully observable Markov decision process*.)

In our example domain, when \mathcal{U} has executed a single instruction incorrectly, it is no longer possible to reach a terminal state with a positive reward. So it might make sense to give the system an action “Abort presentation of instructions” which it could perform after receiving feedback that an error had been made. We assume for our example domain that \mathcal{S} is in fact obligated to present all of the instructions, no matter how \mathcal{U} carries them out. Hence a typical policy might read “Present the instructions stepwise; but if \mathcal{U} makes an error, give all of the remaining instructions in a single bundle (to save time, since there is no chance of completing the task without error anyway).”

For the same combinations of parameters as in Table 3, we computed the optimal policies (using the method of *value iteration*; see, e.g., [3]; [27], chap. 17). Table 4 shows the results. There is no easily understandable relationship between the policies and plans for a given combination of parameters, for two reasons:

1. The availability of feedback has consequences for planning that tend to cancel each other out:
 - On the one hand, \mathcal{S} can plan more dangerously, since it knows that if an error occurs early on, it can at least rush through the rest of the instruction sequence and save time, by presenting instructions in large bundles.
 - On the other hand, there is an advantage in dividing the instructions into smaller groups, since then \mathcal{S} will get its feedback as soon as possible after each instruction.
2. The experimental data from which the probabilities and costs were derived show some patterns whose influence on the planning process is hard to anticipate intuitively. For example, the execution time for the first instruction in a bundle is considerably longer than that for the subsequent instructions in the same bundle, whereas the error rate for the first instruction is much lower.

In sum, one lesson to be drawn from Tables 3 and 4 is that the results of empirically based decision-theoretic planning are not always easy to predict—or, by the same token, to reproduce with simpler methods.

5 OTHER APPROACHES TO DECISION MAKING IN IUIS

To place the preceding discussion into perspective, we will compare the decision-theoretic approach to decision making discussed above with two of the approaches that are most frequently used for IUIs. Although these approaches tend to be implemented in very different ways, Figure 10 represents each of them with a simple influence diagram, so that the high-level relationships can be recognized more easily.

The first diagram simply summarizes more abstractly the model that was introduced in Section 3.

5.1 Content-Based Recommendation

The first alternative type of decision making is found in systems that recommend documents, products, or other objects to \mathcal{U} on the basis of their attributes.

For concreteness, consider first a case where \mathcal{S} ’s action is to recommend exactly one web page that \mathcal{U} might be interested in at the moment. As Figure 10B indicates, one frequent approach is to rate a large number of pages in terms of their relevance with respect to \mathcal{U} ’s information need (as expressed, e.g., in a query). This rating may determine by itself which recommendation is made; or some other ratings may also be taken into account (e.g., the rating of the web page’s size, as in Figure 10B), so that \mathcal{S} ’s decision depends on some combination of the ratings (e.g., a weighted sum).

An analogous approach is followed by product recommendation systems that use multi-attribute utility theory ([32]; [8]) as a basic framework (cf. [15]).¹⁸

To see a first difference between the second and the first diagram, note that there are no variables in the second one that

¹⁸Examples of such systems can be found at the web sites of PersonaLogic (<http://www.personalogic.com/>) and Frictionless Commerce (<http://www.frictionless.com/>).

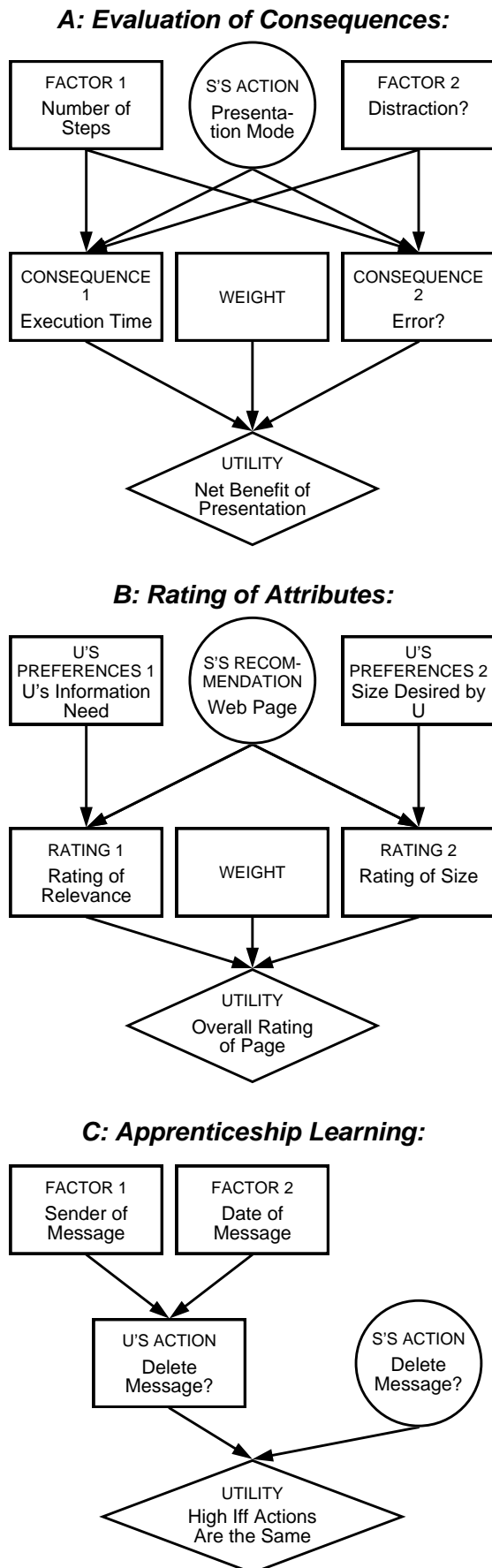


Figure 10. Simplified influence diagrams illustrating some key features of three approaches to decision making in IUIs. (Circles represent decision variables, rectangles chance nodes, and diamonds value nodes. The upper label for each node is an abstract characterization, while the lower label gives a more specific example.)

represent *consequences* of \mathcal{S} 's actions that are *caused* by \mathcal{S} 's decision. Instead, the attributes for which ratings are computed are attributes of the object itself that is being recommended. It is often difficult and unnecessary for \mathcal{S} to anticipate specific consequences of its recommendation of a given document or object for \mathcal{U} . For example, a highly relevant web page may help \mathcal{U} to solve an important problem, or it may simply be interesting for \mathcal{U} to read.

Because it's not a matter of predicting consequences, the decision making process here is largely a matter of arriving at appropriate ratings of objects. When content-based information retrieval is involved, this rating process makes use of information retrieval techniques that are typically sophisticated—and not actually suitable for implementation in the form of an influence diagram. Where simpler attributes of objects are concerned (e.g., the size of a web page; or, in a product recommendation system, the physical size of the product), the main problem is that of determining the user's preferences (e.g., what size is desired) and priorities (e.g., how important size is to \mathcal{U}). Given this information, the rating process is usually straightforward, though a large number of attributes and dimensions may be involved.

In sum, a methodology that involves the empirically based prediction of consequences of system actions will usually not be applicable to the question of which of a large number of objects is most suitable for \mathcal{U} . But note that, once this question has been answered, \mathcal{S} may need to make one or more follow-up decisions, for example: How obtrusively to present the recommendation to \mathcal{U} (if at all); and in what order to present several recommendations. For decisions of these types, it may once again be worthwhile to predict and weigh the objective consequences, such as time costs, disruption of other activities, and the likelihood that \mathcal{U} will pay attention to the recommendation (see, e.g., [21]; [12]; [5]).

Moreover, although it has so far not been usual for systems to predict what will happen if \mathcal{U} follows a recommendation, there are possibilities here that deserve further attention. For example, \mathcal{U} may predictably wind up at the recommended web page, or at the web site of the vendor of the recommended product. Starting from this new state, it may be worthwhile for \mathcal{S} to make further recommendations (e.g., to visit directly linked pages), which would not have been worthwhile before. In other words, the decision-theoretic planning methods that were illustrated in Section 4 are potentially applicable in the area of content-based recommendation, where they can be combined with existing rating techniques (cf. [2]).

5.2 Apprenticeship Learning

A second frequent approach is to have an IUI learn how to make decisions by observing some person (usually the user, but perhaps an expert) who already knows how to make decisions of the type in question. Here are some examples:

- \mathcal{S} learns how \mathcal{U} categorizes and/or deletes emails, so that

- \mathcal{S} can do this work for \mathcal{U} (see, e.g., [20]).
- \mathcal{S} learns how \mathcal{U} makes design choices for a graphical user interface, so that \mathcal{S} can make some of the decisions itself ([9]).
- \mathcal{S} learns from an expert the best way to perform sound equalization, so as to be able to perform the expert’s task for \mathcal{U} ([26]).

Various machine learning techniques have been employed for the learning of this type of model, including decision trees ([24]) and the nearest neighbor method ([26]). To facilitate comparisons, Figure 10C again shows an influence diagram that could in principle be used to realize this approach (although influence diagrams would not in general constitute the most suitable implementation method).

Note that the main job done by an apprenticeship model is the prediction of the action that \mathcal{U} would take. The choice of \mathcal{S} ’S ACTION usually amounts to adopting the most likely action of \mathcal{U} .

A comparison of Figure 10C with Figure 10A shows some salient ways in which the apprenticeship approach differs from our decision-theoretic paradigm:

1. As is shown by the node \mathcal{U} ’S ACTION, in order to learn this type of model \mathcal{S} has to receive feedback on the actions taken by \mathcal{U} in various situations. By contrast, when an action is evaluated on the basis of its objective consequences, the main thing that \mathcal{S} needs to learn about is the consequences. This property is an advantage in cases where there is no-one who could serve as a “teacher”—in particular, where the actions that \mathcal{S} needs to perform are not the same ones that \mathcal{U} might also perform. To illustrate: For the example task discussed in Sections 2–4, there may be no-one in the world who is especially skilled at determining the optimal way of presenting the instructions, taking into account subtle facts about execution times and error rates under different conditions.
2. It is not straightforward to take situation-dependent priorities into account (e.g. the importance of reducing the size of the mailbox, relative to that of preserving information, corresponding to the nodes labeled WEIGHT in Figures 10A and 10B). Any such variable would have to be represented as one of the features that characterize the situations in which \mathcal{U} ’s actions are observed; yet the situation-dependent priority cannot in general be observed directly. By contrast, when what is being learned is the relationship between actions and their consequences in various situations, priority variables do not have to play a role in the learning process.
3. Where the learned model allows for more than one possible action, the choice between them does not take the consequences of actions into account. Suppose, for example, that the decision tree learned by an email apprentice specifies that the user would probably want to delete the current message without reading it (because in 13 out of the 19 similar cases in the past, \mathcal{U} did just this). The decision that is dictated by the model is then to delete the message. The decision making mechanism cannot explicitly take into account the fact

that the consequences of incorrectly deleting a message are in general much more serious than the consequences of incorrectly retaining it; because the mechanism does not refer to the consequences of actions at all.

One response to this problem is to allow \mathcal{U} to set a high threshold of confidence that has to be exceeded before \mathcal{S} can take any action without \mathcal{U} ’s approval (see, e.g., [20]). But this solution doesn’t do justice to the fact that different actions have consequences of different degrees of severity (e.g., the threshold for the autonomous performance of “delete” actions should be higher than the threshold for most other actions, which in turn should not all be equal).

Other systems in this category deal with uncertainty about the correctness of decisions by requiring \mathcal{U} to confirm every system action (see, e.g., [24]) or to choose from among a small set of options suggested by \mathcal{S} (see, e.g., [28]). In these systems, \mathcal{S} is not really making decisions but rather helping \mathcal{U} to make decisions more quickly by making options available which \mathcal{U} might have thought of independently. The main benefit is that \mathcal{U} needs to spend less time searching for the desired option and physically specifying it.

4. Inspection of the learned model does not reveal the *reasons why* the model dictates particular actions under particular circumstances. For example, Eisenstein and Puerta ([9]) point out that the decision trees in their interface generation system are easily readable, in the sense that they make it easy to see what decisions \mathcal{S} will make in what situations: All the reader needs to do to see what decision is dictated in a given situation is to classify that situation using the decision tree. On the other hand, the decision trees do not reveal *why* these particular decisions are appropriate. With influence diagrams, we have more or less the opposite situation: The graphical representation of an influence diagram will not immediately reveal what decisions will be made in a given situations. On the other hand, an influence diagram does make it clear which consequences of actions are taken into account in the making of a decision. Moreover, it is more or less straightforward (depending on the software used) to generate a summary of the decision rules that are implied by the influence diagram, such as the one shown in Table 2.

As was the case with content-based recommendation, it may in some cases be worthwhile to combine the apprenticeship approach and the decision-theoretic approach so as to benefit from their complementary strengths:

- With the apprenticeship approach, a model is learned that yields, for each situation, a set of reasonable candidate actions—essentially, the ones that the user (or expert) *might* perform in that situation.
- The decision-theoretic approach is used to select among these candidate actions.

6 ALTERNATIVE WAYS OF LEARNING DECISION-THEORETIC MODELS

In connection with our example system, the decision-theoretic models were based on the data from a controlled experiment. Of course it is not in general feasible to conduct such an experiment for each type of decision that an intelligent user interface needs to make. And even if this is feasible, the experiment is likely to be different from the real situations in which the system is applied. We will therefore consider various alternatives—at first leaving out of consideration the second complication just mentioned.

6.1 What Users' Data Should Be Employed for Learning?

Various ways of learning the required models are discussed at length in [34]. Here, we just summarize a few points in order to put the discussion in the present article into perspective. This discussion refers to the learning of Bayesian networks and influence diagrams, but some of the points are applicable to the learning of other types of models.

The following options are available:

1. *Learn a general model, then use it without further learning in real situations.* Once the initial learning has been done, this method is relatively easy to implement: The learned model can be applied to each new user without further learning. The model can in principle be learned either in a controlled study or on the basis of real usage data. Controlled studies are usually more expensive, but they have some clear benefits: Variables which are unobservable and/or uncontrollable in real usage situations can often be observed and manipulated in a controlled setting (e.g., the variable `DISTRACTION?` in our experiment).

2. *Learn an individual model for each user in a real application situation.* This approach is the one usually followed when machine learning techniques are used to develop adaptive interfaces. In fact, there have been few cases in which models of “users-in-general” have been learned from data.¹⁹ One obvious advantage is that we can skip the often expensive process of collecting and integrating data on a large number of users. And the data about an individual user in a real situation is often more relevant to decisions about that user than data collected about other users in other more or less different situations—especially for tasks where users tend to differ strongly from each other. On the negative side, it may require a lot of observations about a single user to learn a useful model of that user (though this is not always the case—see, e.g., [28]). During this initial learning period, the system will not be able to make very useful decisions.

3. *Learn a general model, adapt it to individual users.* This approach is basically just a combination of the first two approaches (although there are different ways of adjusting a general model to an individual user). The extra overhead of

¹⁹For descriptions of learned Bayesian networks that refer to a population of users, see [1], [19], and [5].

learning the general model may be justified if it is important for a system to be able to start making useful decisions for each new user right away.²⁰ Many systems that perform collaborative filtering (see, e.g., [5]) can be seen as falling into this third category.

We made an initial comparison of these approaches using the data of the experiment described in Section 2: For each of the 24 subjects, a *general* model was learned on the basis of the data for the other 23 subjects, and an *individual* model for that subject was learned from the subject's own data, processed incrementally in random order. We checked how well the performance of each subject could be predicted with either the general model, the individual model, or a combination of the two:

- The general model yielded somewhat useful predictions starting with the very first trial, whereas the individual model performed very poorly at first, having little or no information to go on.
- The combination of the general and the individual model started out performing no better than the general model, but it outperformed the general model increasingly as more individual data were used for learning.
- After all 72 observations for a given subject had been used for learning, the purely individual model was performing about as well as the combination of general and individual models. If more observations on a given subject were available, apparently there would come a point at which it would be best to rely solely on the individual model.

For detailed discussion of these and other results, we must refer to [34]. That paper also discusses the potential advantages of learning causal models of users that include explanatory unobservable variables (e.g., `COGNITIVE LOAD`), as well as ways of dealing with difficulties that arise in such learning (see also [33]).

6.2 How Can Differences Among Situations Be Taken Into Account?

Even more important than differences among users can be differences among the situations in which a system is used. For example, if we learned a good general causal model for the specific situation of our experiment, we might find differences like the following two in a real usage situation:

- The individual task steps might be longer and/or cognitively more demanding (or shorter and/or simpler).
- Different types of situational distractions might arise, which might be less or more distracting than the distraction task studied in our experiment.

We can distinguish two general approaches to dealing with this problem:

²⁰On the basis of this reasoning, Eisenstein and Puerta ([9]) start their learning with a default initial decision tree, which they then allow to adapt to the individual user. A roughly analogous strategy is pursued by speech recognition systems that begin with a speaker-independent model and proceed to adapt to each individual speaker.

1. *Include some learning in the real situation of use.* In particular, the advantages mentioned above of adapting a general model to the individual user become greater if each individual user also operates in a somewhat different situation.
2. *Modify the learned model by hand.* Sometimes it may be possible to modify the originally learned influence diagram by hand on the basis of a theoretical analysis of the changes in the context. To take a clear example where this approach seems attractive, suppose that the only difference between the original context C and the new context C' is that in C' , under stepwise presentation, the operation that U has to perform in order to confirm completion of a step takes about 300 msec longer than it did in C . The only necessary change in the influence diagram would seem to concern the conditional probabilities (in the CPT for the node EXECUTION TIME) that predict the execution time under stepwise presentation. Specifically, the prediction should be increased by $(n - 1)300$ msec, where n is the number of steps in the sequence. Since errors have negligible frequency under stepwise presentation, there is no reason to expect that the frequency of errors would be affected by this change.

Engineering-oriented cognitive models that have been developed in human-computer interaction research, such as the GOMS model and Model Human Processor ([7]) and their descendants, are intended to support this type of prediction. Still, an obvious limitation of this approach is that the consequences of a change in context may not always be predictable on the basis of theoretical analysis alone. Suppose, for example, that it's not the confirmation operation that is lengthened but rather the execution of a typical task step (e.g., instead of just clicking on a button, U now has to adjust the position of a slider). This change will presumably not just lengthen execution times: Under bundled presentation, it will also lengthen the time during which U has to remember the remaining instructions, thereby increasing the error rate. The size of the increase in error rate is hard to predict reliably in the absence of additional data.

7 USING THE DECISION-THEORETIC FRAMEWORK AS A CONCEPTUAL TOOL

The problems mentioned so far will be so serious in many cases that the whole idea of collecting and using empirical data appears infeasible. But even then, it may be useful to apply decision-theoretic concepts, replacing the detailed empirical data with qualitative educated guesses, perhaps based on previous research:

1. Specify the *structure* of the influence diagram that could in principle be learned with empirical data, using whatever information, experience, or theoretical insight you have available.
2. Describe at least qualitatively the relationships that you think exist among the nodes in the influence diagram.
3. Formulate decision rules that seem appropriate in the light of the qualitative analysis.

Along these general lines, Stephanidis, Karagiannidis, and Koumpis ([31]) propose a method for formulating *adaptation rules* within a decision-theoretic framework that is in some ways similar to the one examined in this article. More generally, The use of influence diagrams and related formalisms to help clarify a decision maker's assumptions and values is common practice in the field of decision analysis (see, e.g., [8]).

To illustrate the use of influence diagrams as a conceptual tool, we can consider the three diagrams shown in Figure 11. These diagrams illustrate the potential—and limitations—of attempts to arrive at adaptation policies without detailed empirical data.²¹

7.1 One User Property, One Criterion Variable

Diagram 1 in Figure 11 refers to a training system that must decide whether to explain a particular application (e.g., an email system) with reference to a concrete, *analogical* conceptual model (e.g., one involving a filing-cabinet metaphor) or with reference to an abstract model. Here, the decision to be made is of the simplest possible type, since only one user property and one criterion variable are involved.

The graph to the left of the influence diagram shows a relationship that would call for different situations to be made for different users: a *crossover interaction* such that the analogical model leads to better performance for users with low visual ability, while the abstract model is better for those with high ability. This qualitative relationship is not only a necessary one for justifying adaptation to individual users; it is also sufficient in this case. It is not necessary to know the exact quantitative nature of the interaction. So even if a designer chose not to conduct an empirical study on this question, they could justify the rules just mentioned if they could argue that a crossover interaction must exist.

7.2 Two User Properties, One Criterion Variable

In Diagram 2, a second user property is taken into account as well: the *learning mode*, which may be *concrete* or *abstract*. One might be able to predict on theoretical grounds the crossover interaction that Sein and Bostrom ([29]) found (shown in the graph to the right of the influence diagram). Unfortunately, merely qualitative knowledge of the two interactions shown for Diagram 2 does not provide grounds for formulating an decision rules: It's obvious enough what to do with users who have high visual ability and an abstract learning mode; but what about those who combine high visual ability with a concrete learning mode? Without empirical data on this specific group, one can only guess whether their visual ability or their learning mode should predominate in determining the decision. So theoretically based rules for this type of decision would be partly well-founded and partly essentially arbitrary.

²¹To ensure realism, the first two diagrams are based on a report of an experiment conducted by Sein and Bostrom ([29]).

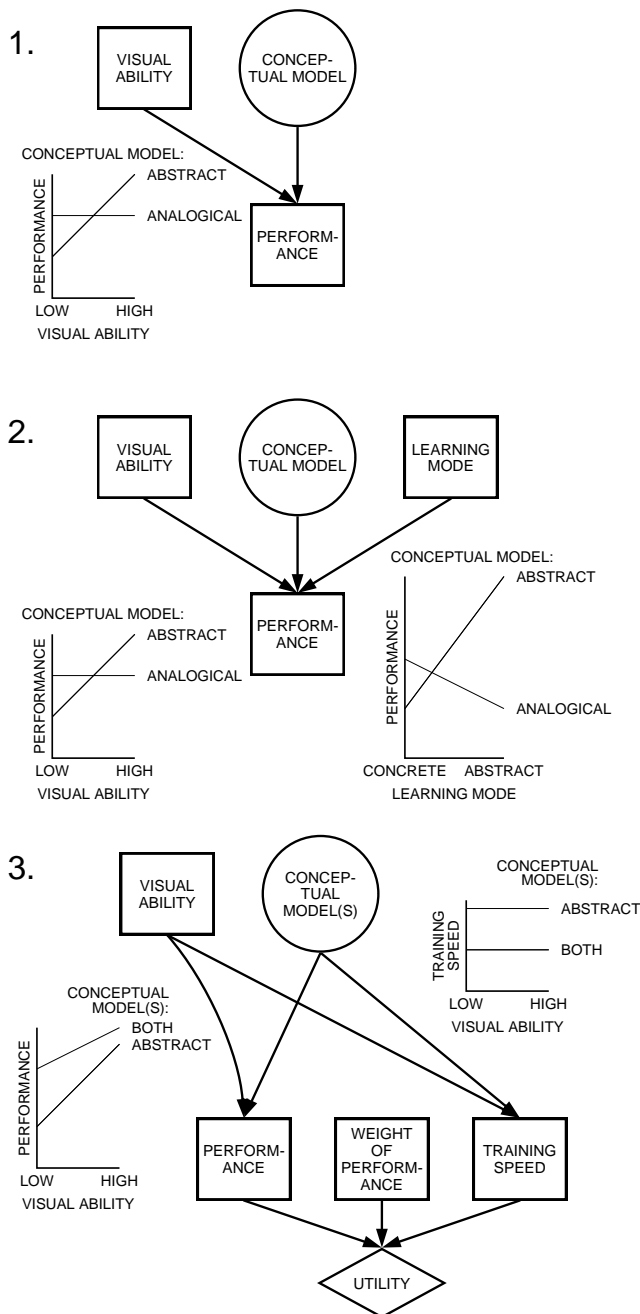


Figure 11. Influence diagrams that might be used for the analysis of three related adaptation decisions. (Explanation in text.)

7.3 One User Property, Two Criterion Variables

In Diagram 3, we suppose that for some reason the only choice available to the system is between (a) presenting only the abstract conceptual model or (b) presenting both the abstract and the analogical model in succession. Although Sein and Bostrom ([29]) didn't investigate this question, we could predict on theoretical grounds that the addition of the analogical model would benefit all users to some extent—especially those with low visual ability. So there's no crossover that in itself would justify making different decisions for different users. Similarly, if we also consider only the second criterion variable of training speed, using only the abstract model is better for all users. The justification for treating users differently can be seen only when the overall value of the decision is considered: For some users, the inclusion of the second model may be worth the decline in training speed, while for others it may not be. So to justify adaptation to individual users, we would have to make some assumption about the relative importance of performance and training speed; and even then the formulation of a policy would involve a good deal of guesswork unless the relevant empirical data were available.

In sum, as the number of relevant variables increases, it becomes increasingly difficult to justify particular adaptation rules solely on the basis of qualitative beliefs about the relationships among the variables—even if it can be assumed that these beliefs are correct. But even if useful empirical data can't be collected, the type of analysis proposed here may help in the formulation of a coherent set of decision rules and in the identification of the aspects of the rules that are most likely to be wrong.

8 CONCLUDING REMARKS

At the present time, decision making methods that evaluate the expected consequences of system actions are seldom employed in intelligent user interfaces (though a number of exceptions have been cited above). One reason is probably that the fine-grained decisions for which they are generally most suitable tend to constitute a small part of any given IUI. Moreover, we have seen that the task of creating a sound empirical basis for the application of such methods is challenging, especially when a wide range of users and contexts is involved.

On the positive side, these methods have strengths that complement those of the more frequently used methods:

- They allow a differentiated consideration of tradeoffs among consequences, taking into account situationally variable priorities.
- They lend themselves to the planning of sequences of related decisions in which each decision takes into account the consequences of the previous ones.
- They don't presuppose the availability of data from a person who already knows how to make the type of decision in question successfully.

- They make the rationale underlying decisions relatively explicit.

Because of these potential benefits, it seems worthwhile for IUI researchers and designers to keep decision-theoretic methods in mind as options which can supplement or replace alternative ways of making decisions. Further research along these lines is likely to produce more efficient methods and tools for acquiring the necessary models and for employing them within the architectures of intelligent user interfaces.

ACKNOWLEDGMENTS

Some of the results and theoretical discussion in the present article were presented in [14]. The research described was supported by the German Science Foundation (DFG) in its Collaborative Research Center on Resource-Adaptive Cognitive Processes (<http://www.coli.uni-sb.de/sfb378/>), SFB 378, Projects A2 (VEVIAG) and B2 (READY).

REFERENCES

1. D. W. Albrecht, I. Zukerman, and A. E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8:5–47, 1998.
2. T. Bohnenberger and A. Butz. Decision-theoretic planning of navigation instructions. In *ECAI 2000 Workshop on Artificial Intelligence in Mobile Systems*, Berlin, 2000.
3. C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence*, 11:1–94, 1999.
4. S. Bradshaw, A. Scheinkman, and K. Hammond. Guiding people to information: Providing an interface to a digital library using reference as a basis for indexing. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*. ACM, New York, 2000.
5. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, page 43–52. Morgan Kaufmann, San Francisco, 1998.
6. W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.
7. S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ, 1983.
8. R. T. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis*. Duxbury, Pacific Grove, CA, 1996.
9. J. Eisenstein and A. Puerta. Adaption in automated user-interface design. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*, page 74–81. ACM, New York, 2000.
10. B. Großmann-Hutter, A. Jameson, and F. Wittig. Learning Bayesian networks with hidden variables for user modeling. In *Proceedings of the IJCAI99 Workshop “Learning About Users”*, Stockholm, 1999.
11. E. Horvitz. Principles of mixed-initiative user interfaces. In M. G. Williams, M. W. Altom, K. Ehrlich, and W. Newman, editors, *Human Factors in Computing Systems: CHI '99 Conference Proceedings*, page 159–166. ACM, New York, 1999.
12. E. Horvitz, A. Jacobs, and D. Hovel. Attention-sensitive alerting. In K. B. Laskey and H. Prade, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference*, page 305–313. Morgan Kaufmann, San Francisco, 1999.
13. E. Horvitz and T. Paek. Conversation as action under uncertainty. In C. Boutilier and M. Goldszmidt, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference*. Morgan Kaufmann, San Francisco, 2000.
14. A. Jameson, B. Großmann-Hutter, L. March, and R. Rummer. Creating an empirical basis for adaptation decisions. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*, page 149–156. ACM, New York, 2000.
15. A. Jameson, R. Schäfer, J. Simons, and T. Weis. Adaptive provision of evaluation-oriented information: Tasks and techniques. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, page 1886–1893. Morgan Kaufmann, San Mateo, CA, 1995.
16. A. Jameson, R. Schäfer, T. Weis, A. Berthold, and T. Weyrath. Making systems sensitive to the user’s changing resource limitations. *Knowledge-Based Systems*, 12:413–425, 1999.
17. F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In R. Lopez de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, page 367–373. Morgan Kaufmann, San Francisco, 1994.
18. S. Kerpedjiev and S. F. Roth. Mapping communicative goals into conceptual tasks to generate graphics in discourse. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*, page 157–164. ACM, New York, 2000.
19. T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling Web query dynamics. In J. Kay, editor, *UM99, User Modeling: Proceedings of the Seventh International Conference*, page 119–128. Springer Wien New York, Vienna, New York, 1999.

20. P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.
21. P. P. Maglio, R. Barrett, C. S. Campbell, and T. Selker. SUITOR: An attentive information system. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*, page 169–176. ACM, New York, 2000.
22. L. March. Ressourcenadaptive Instruktionen in einem Hotline-Szenario [Resource-adaptive instructions in a hotline scenario]. Master’s thesis, Department of Psychology, Saarland University, Germany, 1999.
23. M. T. Maybury and W. Wahlster. Intelligent user interfaces: An introduction. In M. T. Maybury and W. Wahlster, editors, *Readings in Intelligent Interfaces*, page 1–13. Morgan Kaufmann, San Francisco, CA, 1998.
24. T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, 1994.
25. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.
26. D. Reed. A perceptual assistant to do sound equalization. In H. Lieberman, editor, *IUI2000: International Conference on Intelligent User Interfaces*, page 212–218. ACM, New York, 2000.
27. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
28. R. B. Segal and J. O. Kephart. Incremental learning in SwiftFile. In P. Langley, editor, *Proceedings of the 2000 International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, 2000.
29. M. K. Sein and R. P. Bostrom. Individual differences and conceptual models in training novice users. *Human-Computer Interaction*, 4:197–229, 1989.
30. R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.
31. C. Stephanidis, C. Karagiannidis, and A. Koumpis. Decision making in intelligent user interfaces. In J. D. Moore, editor, *IUI97: International Conference on Intelligent User Interfaces*, page 195–202. ACM, New York, 1997.
32. D. v. Winterfeldt and W. Edwards. *Decision Analysis and Behavioral Research*. Cambridge University Press, Cambridge, England, 1986.
33. F. Wittig and A. Jameson. Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks. In C. Boutilier and M. Goldszmidt, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference*. Morgan Kaufmann, San Francisco, 2000.
34. F. Wittig and A. Jameson. Learning general models of users for adaptive interfaces. Manuscript submitted for publication, 2000.