In Henry Lieberman (Ed.), *IUI 2000: International Conference on Intelligent User Interfaces*, (pp. 149–156) New York: ACM.
(Also available from http://w5.cs.uni-sb.de/~ready/.)

Creating an Empirical Basis for Adaptation Decisions

Anthony Jameson Barbara Großmann-Hutter Leonie March Ralf Rummer
Department of Computer Science / Department of Psychology, University of Saarbrücken
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
Fax: +49 681 302-4136, Phone: +49 681 302-2363 / 302-3196

1 ABSTRACT

How can an adaptive intelligent interface decide what particular action to perform in a given situation, as a function of perceived properties of the user and the situation? Ideally, such decisions should be made on the basis of an empirically derived causal model. In this paper we show how such a model can be constructed given an appropriately limited system and domain: On the basis of data from a controlled experiment, an influence diagram for making adaptation decisions is learned automatically. We then discuss why this method will often be infeasible in practice, and how parts of the method can nonetheless be used to create a more solid basis for adaptation decisions.

1.1 Enumerate Keywords

Adaptive systems, Experiments, Decision theory, Influence diagrams, Bayesian networks

2 INTRODUCTION

One way in which an intelligent user interface can be intelligent is by adapting autonomously to properties of the user or the situation. A *user-adaptive system* can be defined as a system ($\mathcal{S}$) that (a) makes nontrivial inferences concerning properties of the user ($\mathcal{U}$) on the basis of information about $\mathcal{U}$ and (b) adapts its actions to the inferred properties of $\mathcal{U}$ and relevant contextual factors (cf. [6]).

How do user-adaptive systems go about choosing their actions? In descriptions of such systems, the decision procedures are usually described—if at all—in terms of if-then rules or formulas that are accompanied by little empirical or theoretical justification. Even with systems that employ explicit decision-theoretic methods (see, e.g., [5]; INTRODUCTION[7]; [13]), the empirical basis of the decision procedure is typically not in the focus of attention.

In this paper, we aim to encourage and help designers of user-adaptive systems to develop decision procedures in a more principled and—if possible—empirically justified way. We address two questions in turn:

1. In an ideal situation—with a simple, restricted system about which we can collect any empirical data we like—what would be an effective method for developing an optimal decision procedure?
2. In the real world—in which the method just introduced is normally infeasible—how can we adopt some aspects of the method in order at least to improve on current practice?

To answer the first question, in the next section we introduce a simple system and an experiment that we performed with it.

3 EXAMPLE DOMAIN AND EXPERIMENT

3.1 The Specific Adaptation Issue

The example system ($\mathcal{S}$) and domain used in our experiment are illustrated in Figure 1. $\mathcal{S}$ is an assistance system that presents sequences of spoken instructions to the user $\mathcal{U}$ (as, for example, a computer support hot-line might do). One question that arises is whether $\mathcal{S}$ should present the instructions (a) in a *stepwise* manner (i.e., allowing $\mathcal{U}$ to execute each instruction in the sequence before hearing the next one) or (b) in a *bundled* manner (i.e., all at once, before $\mathcal{U}$ starts executing the first instruction) (cf. Figure 2).

The main drawback of stepwise presentation is the *interaction overhead*: After executing each instruction (except the last one in the sequence), $\mathcal{U}$ must somehow confirm to $\mathcal{S}$ that he is ready for the next instruction. (We are assuming here that $\mathcal{S}$ does not get direct information about $\mathcal{U}$'s task performance.) This confirmation signaling requires a certain amount of time and effort on $\mathcal{U}$'s part.[1]

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 remem-

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

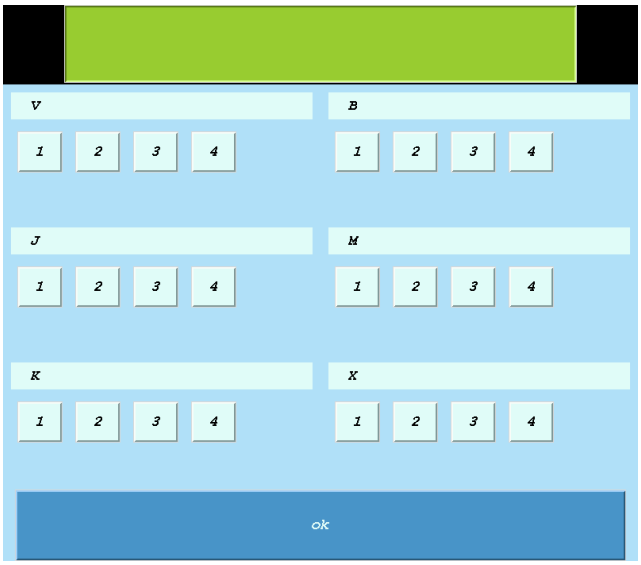

Figure 1. Main screen used for the experiment.

Stepwise:	*Bundled:*
$\mathcal{S}$: Set *X* to *3*.	$\mathcal{S}$: Set *X* to *3*, set *M* to *1*, set *V* to *4*.
$\mathcal{U}$: ... [OK]	$\mathcal{U}$: ... ... [OK]
$\mathcal{S}$: Set *M* to *1*.	
$\mathcal{U}$: ... [OK]	
$\mathcal{S}$: Set *V* to *4*.	
$\mathcal{U}$: ... [OK]	

Figure 2. Illustration of the two presentation modes for instructions.

ber 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.

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.

3.2 Method

3.2.1 Materials

Figure 1 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 2).[2] 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.

3.2.2 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:[3]

- 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.[4] Specifically: In bundled mode, this was the duration of the interval between (a) the moment the system finished playing the instruction sequence and (a) 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, whereby the time required to press the *OK* button (not relevant in bundled mode) was also included in the execution time.
- Error
  This variable had the value 0 for a particular instruction

[2] The original German formulation of an instruction was "Setze [letter] auf [number]".

[3] Further dependent variables are analyzed in [10], but the results are not relevant enough here to warrant discussion in this paper.

[4] Note 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.

sequence if the subject pressed all of the instructed buttons in the correct order, the value 1 otherwise.

3.2.3 Subjects

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

3.2.4 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 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.

3.3 Results

Most interesting for our purposes are the results for the 12 specific combinations of independent variables, shown in Figure 3. 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.

For each of the two dependent variables, a three-way ($3 \times 2 \times 2$) analysis of variance (ANOVA) was conducted.[5]

3.3.1 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; if the presentation is stepwise; and if there is a distraction task.[6]

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 sequence length and presentation mode.

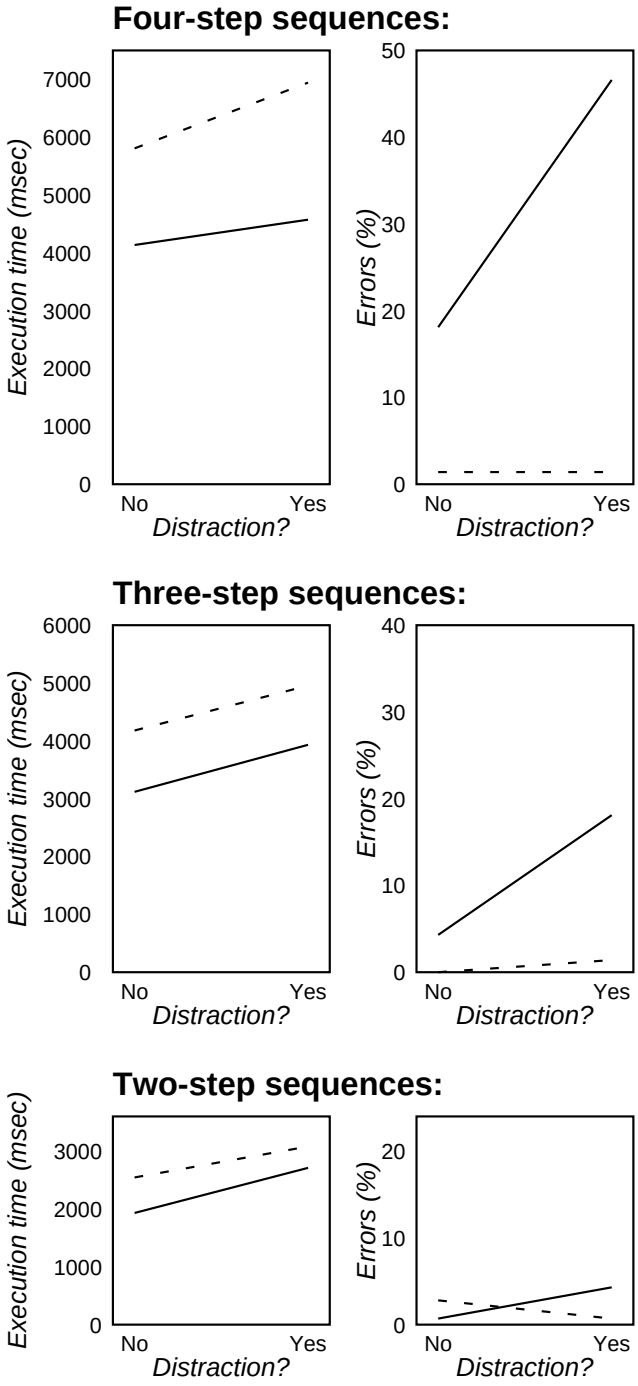

Figure 3. Mean execution times and error rates for each combination of values of the independent variables.
(Dashed lines represent stepwise presentation, solid lines bundled presentation.)

[5] Previously conducted multivariate analyses of variance had confirmed that it was appropriate to conduct these separate ANOVAs.

[6] Except where stated otherwise, all reported differences are statistically significant at least at the level of $p < .01$.

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 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.

3.3.2 *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; if the presentation is bundled; and if there is a distraction task.

The ANOVAs also revealed several two-way interactions involving pairs of independent variables; but since these are most easily interpretable in terms of results for specific combinations of three independent variables, we will not discuss them.

Most relevant for our purposes is the significant three-way interaction of the independent variables. The most salient specific features of the results are the high error rates under bundled presentation when there is a distraction task (except with two-step sequences, where the error rates are too low to be of much interest). For four-step sequences (see the top right-hand graph in Figure 3), a Scheffé test shows that this error rate is significantly higher than the other three error rates shown. For three-step sequences, the corresponding error rate is significantly higher than the lowest error rate in the same graph ($p = .02$), but the differences from the other two error rates are not significant ($p = .065$ and $p = .287$, respectively).

In sum, even though the last two differences do not (quite) reach accepted levels of statistical significance, the analyses of variance show that the overall pattern of results for errors represents stable underlying regularities that can be expected to reoccur in similar situations.

3.4 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.

4 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 adaptation decisions. 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.

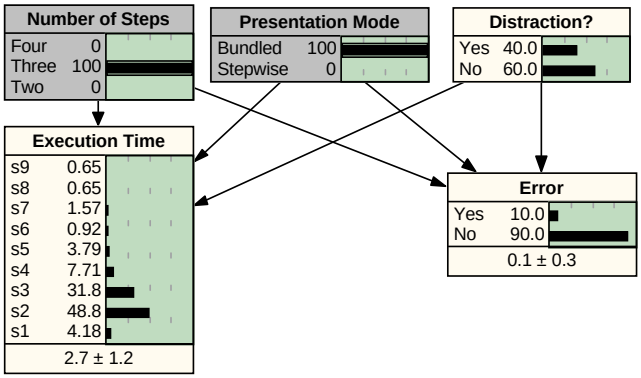

Figure 4. Bayesian network learned on the basis of the experimental data, showing a prediction made under uncertainty about the independent variable Distraction.

4.1 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) with the structure shown in Figure 4. 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 Bayesian network that contains only observable variables and whose structure has been specified in advance is straightforward.[7] We used the NETICA[8] built-in-algorithm for learning BNs. This algorithm, which presupposes that all variables are observable, computes the (conditional) probabilities on the basis of frequencies in the data.[9]

The properties of the data that are summarized in Figure 3 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:

38. *Predicting the dependent variables given uncertainty about the values of the independent variables:* $\mathcal{S}$ may want to make a prediction (and an adaptation decision) even without knowing the values of all of the independent variables shown in Figure 4. For example, $\mathcal{S}$ may not know whether

[7] In a modeling effort that is in some ways comparable to the present one, Lau and Horvitz ([9]) used data on users' WWW search behavior to learn BN models that can be used to predict and interpret such behavior.

[8] NETICA is a commercial tool from Norsys Software Corp. (see http://www.norsys.com) for working with Bayesian networks and influence diagrams.

[9] We 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 [3]). To focus attention on the central methodological issues, we discuss here only the simplest possible network that could be defined for this experiment.

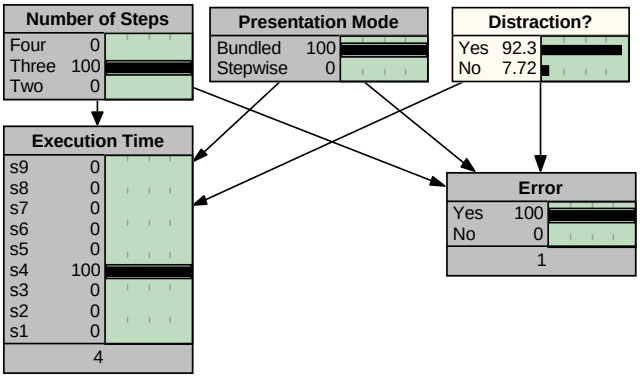

Figure 5. The same network as in Figure 4, showing the interpretation of an observation of $\mathcal{U}$'s performance.

$\mathcal{U}$ is currently distracted. As Figure 4 shows, if a probability distribution for each of these variables is specified, the network will still generate a prediction for $\mathcal{U}$'s execution time.

39. *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 uncertainty. For example, Figure 5 shows how the network infers that $\mathcal{S}$ was probably distracted after observing that $\mathcal{U}$ made an error. This learning about the current $\mathcal{U}$ can enhance the quality of $\mathcal{S}$'s future predictions and decisions about this particular $\mathcal{U}$.[10] 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.

4.2 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 6), in two steps:

1. Add a *value node* that expresses the system's evaluation of a particular combination of values of the dependent 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 chance variable Weight of Error. $\mathcal{S}$ can instantiate this variable to various values to take into account shifting priorities. For example, a value of 16 means that avoiding 1 error is just as important as saving 16 seconds of execution time.

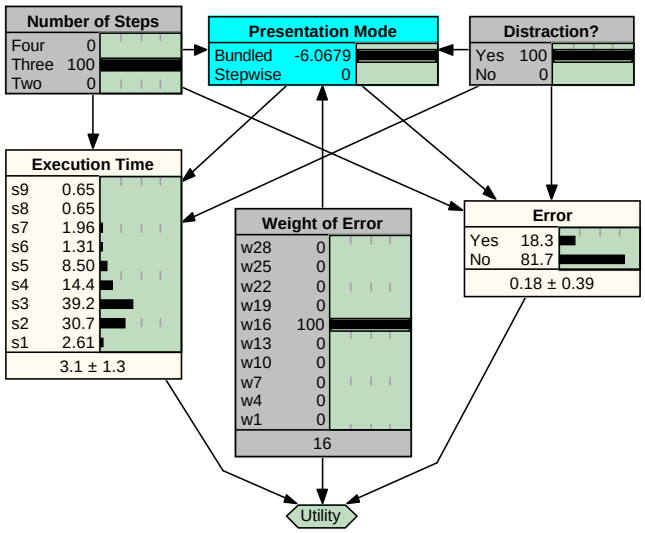

Figure 6. An influence diagram defined as an extension to the BN of the previous figures.

2. Make Presentation Mode into a *decision node*. The three links pointing into this node from other variables express the fact that the values of these other variables may be known exactly at the time when a decision is made. Once the complete influence diagram has been specified and solved,[11] $\mathcal{S}$ knows the utility of each action it might take in each situation. For example, Figure 6 shows that the utility of bundled presentation in the situation described by the other nodes is -6.0679. By comparing this utility with that of stepwise presentation, $\mathcal{S}$ can decide which mode to use.

In addition to having $\mathcal{S}$ make decisions in individual cases, the designer will probably want to have an overall picture of the decisions that $\mathcal{S}$ will make. For example, it's conceivable that, for any reasonable value of Weight of Error, $\mathcal{S}$ 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. Tools for evaluating influence diagrams 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 *policy* 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 policy for the present influence diagram is summarized concisely in Table 1 on the basis of the actual policy table generated automatically for the influence diagram.

Note that the type of information provided by the policy for a decision node can have a similar function to the results of the sensitivity analyses that are sometimes performed with pre-

[10] If 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., [7].

[11] For discussions of solution algorithms, see [12] and [8]; the algorithm used by NETICA is based on the latter work.

Table 1. Summary of the policy for the decision node Presentation Mode

Steps	Distraction = "No"	Distraction = "Yes"
Four	Stepwise iff $w > 9$	Stepwise iff $w > 3$
Three	Stepwise iff $w > 21$	Stepwise iff $w > 6$
Two	Always bundled	Stepwise iff $w > 9$

dictive models of interface designs. For example, [1] (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 existing methods. It is a convenient feature of 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 idealized method just proposed is straightforward, in that it combines well-known methods from experimental psychology with some of the simpler functions of readily available decision-theoretic software tools. Yet it permits a more effective use of the experimental data for adaptation decisions than would be possible through the use of a conventional data analysis.

5 PARTIAL APPLICATIONS OF THE IDEALIZED METHODOLOGY

Unfortunately, this methodology will be infeasible in most practical situations. The main problem is that it is usually impossible to obtain experimental data in a situation that is identical to the situations in which the system is actually to be used.

But against the background of the discussion so far, we can identify several fallbacks that enable the designer to derive adaptation policies in a principled way, possibly with an empirical basis.

5.1 Fallback 1: Modify the Learned Model by Hand

In a relatively favorable situation, we may have developed a model like the one described above and want to apply it to a system or domain that differs in just a few specific ways from the one for which the experimental data were collected.

Here are examples of relatively simple differences that might arise:

- 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.

Obviously, if exactly the same decision procedure is used in the new situation, the decisions will in general be based on false premises and therefore often be inappropriate.

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 $\mathcal{U}$ 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 ([1]) 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, $\mathcal{U}$ now has to adjust the position of a slider). This change will presumably not just lengthen execution times: Under blocked presentation, it will also lengthen the time during which $\mathcal{U}$ 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.

5.2 Fallback 2: Collect Real Usage Data

A designer might think that any controlled experiment would be so far removed from the reality of actual system use that even the sort of theoretical extrapolation just discussed would not yield a useful adaptation policy. Or there may simply be insufficient resources available for a controlled experiment.

An alternative may be to employ data on actual system use instead. While data are being collected, $\mathcal{S}$ might be acting nonadaptively or applying some suboptimal adaptation policy. But this type of data is likely to have some important limitations relative to experimental data, including various types of missing data, uncontrolled contextual factors, and bias introduced by whatever decision policy $\mathcal{S}$ is applying during the learning phase. Even if statistical learning methods are employed that deal as well as possible with such complications (see, e.g., [4]), it may be impossible to learn a BN which is accurate enough to be useful.

5.3 Fallback 3: Do the Analysis Without Data

The problems mentioned so far will be so serious in many cases that the whole idea of collecting empirical data and learning from it appears to be infeasible. But even then, it may be useful to apply the same basic logic of the idealized methodology, replacing the empirical data with qualitative

educated guesses:

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 a policy that seems appropriate in the light of the qualitative analysis.

The three influence diagrams shown in Figure 7 are typical of the kinds that a designer might be able to draw without the benefit of empirical data, as a way of making explicit his or her beliefs.[12] They illustrate the potential—and limitations—of attempts to arrive at adaptation policies without empirical data.[13]

5.3.1 1. One User Property, One Criterion Variable

Diagram 1 shows the case of a training system that must decide whether to explain a particular application (e.g., an e-mail 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 adaptation decision is 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 justify adaptation: 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, 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 policy just mentioned if they could argue that a crossover interaction must exist.

5.3.2 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 ([11]) 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 adaptation policy: 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 a theoretically based policy for this type of decision would be partly well-founded and partly essentially arbitrary.

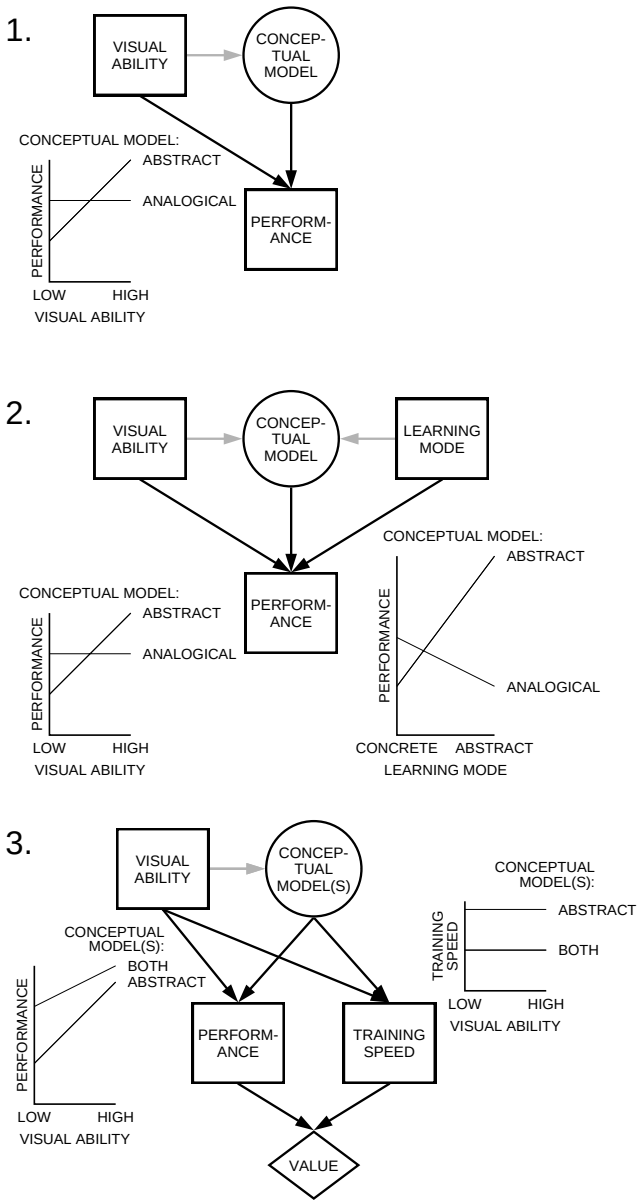

Figure 7. Illustrative influence diagrams.
(Explanation in text.)

5.3.3 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 ([11]) 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 any adaptation. Similarly, if we also consider only the second criterion variable of training speed, us-

[12] 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., [2]).

[13] Actually, to ensure realism, the first two diagrams are based on an experiment of Sein and Bostrom ([11]).

ing 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, 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 a particular adaptation policy 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 policy and in the identification of the aspects of the policy that are most likely to be wrong.

6 SUMMARY OF CONTRIBUTIONS

The content-specific message of this paper concerns some tradeoffs that need to be taken into account when a system decides how to present a sequence of instructions to a user.

The more general message is methodological. In showing how designers might in principle create a solid empirical basis for adaptation decisions, we have mainly succeeded in showing how difficult it usually is to do so in practice. But still, we hope to have contributed a clearer understanding of the problem, so that designers (a) "know what they don't know" when considering adaptation policies and (b) can take whatever steps are practically feasible in their particular situation toward deriving a solid basis for their adaptation policies.

7 ACKNOWLEDGMENTS

This research was supported by the German Science Foundation (DFG) in its Collaborative Research Center on Resource-Adaptive Cognitive Processes, SFB 378, Projects A2 (VEVIAG) and B2 (READY). The learning of the Bayesian networks was prepared by Frank Wittig. The final version benefited considerably from the comments of two anonymous reviewers.

REFERENCES

1. S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ, 1983.

2. R. T. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis*. Duxbury, Pacific Grove, CA, 1996.

3. 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.

4. D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1998.

5. 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*, pages 159–166. ACM, New York, 1999.

6. A. Jameson. User-adaptive systems: An integrative overview. Tutorial presented at UM99, the Seventh International Conference on User Modeling, Banff, Canada, 1999. Available from http://www.cs.uni-sb.de/users/jameson/.

7. A. Jameson, R. Schäfer, T. Weis, A. Berthold, and T. Weyrath. Making systems sensitive to the user's time and working memory constraints. In M. T. Maybury, editor, *IUI99: International Conference on Intelligent User Interfaces*, pages 79–86. ACM, New York, 1999.

8. 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*, pages 367–373. Morgan Kaufmann, San Francisco, 1994.

9. 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*, pages 119–128. Springer Wien New York, Vienna, New York, 1999.

10. L. March. Ressourcenadaptive Instruktionen in einem Hotline-Szenario [resource-adaptive instructions in a hotline scenario]. Master's thesis, Department of Psychology, 1999.

11. M. K. Sein and R. P. Bostrom. Individual differences and conceptual models in training novice users. *Human-Computer Interaction*, 4:197–229, 1989.

12. R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

13. D. Suryadi and P. J. Gmytrasiewicz. Learning models of other agents using influence diagrams. In J. Kay, editor, *UM99, User Modeling: Proceedings of the Seventh International Conference*, pages 223–232. Springer Wien New York, Vienna, New York, 1999.