The Experimental Study of Adaptive User Interfaces

Pat Langley (langley@isle.org)
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306

Michael Fehling (fehling@odc.stanford.edu)
Organizational Dynamics Center, EES/OR Department
Stanford University, Stanford, CA 94305

Abstract

In this paper we examine some issues that arise in the experimental evaluation of adaptive user interfaces, which are computational decision aids that use machine learning to improve their interaction with users. We begin by reviewing the notion of an adaptive interface and presenting examples for a number of application areas. After this, we discuss plausible dependent measures of their behavior, including solution speed, solution quality, user satisfaction, and predictive accuracy. Next we turn to independent variables that are likely to influence these measures, including the number of user interactions and characteristics of the system, the user, and the task at hand. In closing, we comment on the role that experimentation plays in the larger scientific process.

This work was funded by the Office of Naval Research under Grant N00014-96-1-1221. We thank Nicolas Fiechter, Wayne Iba, Seth Rogers, and Stephanie Sage for their comments on an earlier draft of the paper.

1. The Promise of Adaptive User Interfaces

As computers have become more widespread, the software that runs on them has also become more interactive and responsive. The days of programming on punch cards and submitting overnight jobs are remembered by only a few early users, and even the era of time-sharing systems and text editors has become a dim memory. Modern operating systems support a wide range of interactive software, from WYSIWYG editors to spreadsheets to computer games, most embedded in some form of graphical user interface. Such packages have become a central part of business and academic life, with millions of users dependent on them for accomplishing their daily goals.

Another type of interactive software, which developed independently of graphical interfaces, took the form of computational decision aids or advisory systems. These were typically knowledge-based systems designed to help human decision makers in a specific domain like medical diagnosis, stock selection, or military planning. Early decision aids relied on textual interfaces rather than graphical ones, but interaction with the user was always central to their operation.

A recurring problem with both types of interactive software is that they have little flexibility in their behavior, and thus cannot respond to demands from different users. One can modify the default parameter settings on most editors and alter the difficulty level on many computer games, but incorporating more subtle preferences is beyond the scope of most interactive systems, especially computational aids designed to support complex decision making. Even when the user can customize the system's behavior, the standard approach involves filling out an online form; this may reflect the user's beliefs but not his actual practices, for even domain experts have difficulty describing their true knowledge and biases about a domain.

However, in recent years, a new approach to flexible software has emerged under the rubric of *adaptive user interfaces* (Langley, 1997). These systems are computational decision aids that interact with a user and, based on their experience with that person, improve their ability to serve his needs over time. The typical adaptive interface accomplishes this by providing advice to the user, collecting traces of user decisions about that advice, constructing a user profile from these traces through machine learning, and using this profile to alter its future interactions. In this way, the resulting advisory system is *personalized* to the individual user.

A number of key issues arise in the design of adaptive user interfaces. One concern is that the process of collecting feedback should not place excessive demands on the user. Thus, a system that collects decision traces as part of its normal interaction is preferable to one that requires users to explicitly rate alternatives. In addition, adaptive interfaces benefit from the use of online learning, which updates the user model after every interaction, and from rapid learning, since users will typically expect the system to reflect their preferences soon after they start using it. Both concerns contrast with the typical emphases in machine learning, which focuses on offline learning and obtaining high asymptotic performance.

In summary, adaptive user interfaces constitute a relatively new type of artifact that deserve closer study. As Simon (1969) has argued, sufficiently complex artifacts are best understood not through formal analysis but through the same experimental method that predominates in the natural sciences. In the last decade, such experimentation has become common within artificial intelligence, especially in machine learning, and many of the concepts and techniques from the study of such intelligent artifacts carries over to the study of adaptive user interfaces. Nevertheless, they differ from previous software entities in some important ways, which suggests a careful examination of the issues surrounding their experimental evaluation.

In this paper, we discuss the experimental study of adaptive user interfaces. The goal of scientific experimentation in any domain is to better understand a class of behaviors and the conditions under which they occur. Ideally, this understanding will lead to empirical laws and theories, as well as to tests of those theories. As normally defined, an experiment involves systematically varying one or more independent variables and examining their effect on some dependent variables. Thus, an experiment with an adaptive interface requires more than one observation of a system's behavior; it requires a number of observations made under different conditions. In each case, the experimenter must measure some aspect of the system's behavior for comparison across different conditions.

We begin with a brief review of some adaptive user interfaces, which we use later to illustrate important concepts. After this, we examine some dependent measures that make sense for the experimental study of such systems, then discuss four broad classes of independent variables that seem likely to affect these measures. Because adaptive interfaces borrow elements from machine learning, cognitive psychology, and human-computer interaction, these variables should be familiar to researchers from those fields. We close the paper with comments about the broader context in which experiments occur and their role in scientific progress. Many of our observations will hold for the study of computational decision aids in general, but they are still worth stating for the new class of adaptive artifacts.

2. Examples of Adaptive User Interfaces

Before examining the issues that arise in experiments with adaptive interfaces, we should first consider some examples. There are many possible applications for such advisory systems, but one very common use concerns the task of *information filtering*, in which the aim is to provide the user with material that he will find informative or useful. Software of this sort has a long history in information retrieval, a field with a strong tradition of empirical evaluation, but the recent focus on this topic has been driven by the growing popularity of the World Wide Web. There now exist a number of information-filtering systems that incorporate user feedback and adapt to user preferences in response.

One example interface is Pazzani and Billsus' (1997) Syskill & Webert, which recommends Web pages on a given topic that the user is likely to find interesting. Starting from a handcrafted page for the topic, the user marks suggested pages as desirable or undesirable, and the system uses this feedback as training data to develop a model of his preferences. Syskill & Webert represents each user profile as a naive Bayesian classifier, which stores a conditional probability distribution over a set of predictive features, in this case words that occur in the Web page. The system invokes this user profile and compares it with the words in a candidate document when deciding whether to recommend that document to the user, which biases it toward documents that are similar to ones the user has previously ranked highly.

Although recommending Web pages is a common application of adaptive information filtering, other uses are also possible. Another system, Lang's (1995) NewsWeeder, recommends news stories to readers, again using the words in each story to predict whether the user will find it interesting. Another popular task involves sorting and prioritizing electronic mail, typically using words that occur in the message headers and body (e.g., Boone, 1998). A quite different but still common technique recommends items that the user might enjoy, based on that user's ratings and the ratings from other users with similar profiles. Amazon.com draws on such a *collaborative filtering* mechanism to recommend books to its customers.

Another class of adaptive user interfaces address more complex tasks that involve generating new knowledge structures to satisfy the user's goals. For example, Hermens and Schlimmer (1994) describe an adaptive system of this sort for filling out repetitive forms. Their interface suggests values for various fields in the form, but these are defaults that the user can always override. Once the user completes the form, the system interprets the entries as opportunities for learning and uses them to revise its existing predictive rules. Each such rule specifies a default value for a given field based on fields earlier in the form and those in previous forms. Although this work did not focus on personalization per se, Schlimmer and Hermens (1993) incorporated a very similar approach in their personalized interface for note taking. This adaptive system learns a grammar that predicts the order and content of a user's notes, aiming to reduce keystrokes and to help them organize their thoughts.

Rogers and Langley (1998) report another adaptive user interface with a generative flavor, this one focusing on giving advice to drivers. Their Adaptive Route Advisor accepts a current and desired location from the user, carries out best-first search through a digital map to find a few high-quality routes, and presents the user with these options. When the user accepts one of the suggested routes, the system incorporates this decision into its training set and revises its user model, which it represents as relative weights on global route features like the number of turns, the distance, the number of intersections, and the estimated driving time. The algorithm that updates this user model carries out a hill-climing search through the weight space, aiming for parameters that summarize past choices the user has made. The system then draws on the revised model to direct search for routes on future tasks.

Yet another adaptive interface, this one described by Iba, Gervasio, and Langley (1998), focuses on scheduling in the domain of chemical spills and fires. Their INCA system retrieves a schedule from a case library that best matches the features of the current incident, then lets the user interactively modify them for application to that situation, with the system suggesting likely repairs. Once the user decides on an acceptable schedule, INCA passes this solution on to an execution module, which may lead to new events and the need for further repairs to the schedule. Personalization occurs through storage in the case library of final schedules, which presumably reflect user preferences about desirable solutions, and through induction of rules about the conditions under which the user makes each type of repair. INCA uses the expanded case library on future problems and uses its revised repair model to recommend future revisions.

This list does not exhaust the work on adaptive user interfaces, which is an active area with many ongoing research efforts. However, it should clarify the range of tasks to which one can apply this basic idea and it should motivate our discussion of experimental methods. We can now consider in more detail the issues that arise in the experimental evaluation of adaptive interfaces, drawing on example studies from the systems we have already considered.

3. Dependent Measures

Experimental studies in nearly every discipline concern some form of *behavior*. This means they require some measure of that behavior to play the role of a dependent variable in the experiment, that is, the variable that is affected by the experimental manipulations. Different dependent variables make sense for different fields, but here we consider four general types of measures that seem appropriate for the study of adaptive interfaces.

Table 1. Evaluation of INCA's interactive scheduler in terms of (a) solution time and (b) solution quality under three different experimental conditions (Iba et al., 1998).

	(a) SOLUTION TIME	(b) SOLUTION QUALITY
GENERATED FROM SCRATCH	168.98 ± 17.07	33.67 ± 4.06
GENERATED AND REPAIRED	203.27 ± 30.88	29.52 ± 4.42
RETRIEVED AND REPAIRED	127.35 ± 19.91	34.33 ± 4.64

3.1 Measures of Efficiency

People typically invoke computational decision aids, including adaptive user interfaces, because they expect the software will let them accomplish some task more rapidly and with less effort than they can do on their own. This makes the *efficiency* of decision making or problem solving an obvious dependent variable to use when evaluating such an adaptive interface. However, one can instantiate this metric in different ways, each of which reflects only part of the picture.

One natural measure of efficiency is the time the user takes to complete his interaction with the advisory system. For example, Table 1 (a) shows results from an experimental study with INCA, the interactive scheduler described earlier, that compared user behavior under various conditions. One version of the system presented the user with an empty schedule, another used heuristic search to generate the initial schedule which the user then repaired, and a third version retrieved a schedule from its case library for revision by the user. The dependent measure was the number of seconds taken to transform this initial schedule into one the user found acceptable.

But time is not the only measure of efficiency; another facet is the effort that the user must exert to make a decision or solve a problem. Here the most obvious metrics concern the number of user actions that occur during solution of a given problem. In evaluating their system for aiding the completion of repetitive forms, Hermens and Schlimmer measured the number of keystrokes that the user took to complete the form, which they found generally decreased over time as the user interacted with the system. Keystrokes were the obvious performance measure for this interface, but mouse clicks would be more appropriate for an adaptive graphics package and utterances would be natural for any system that incorporates a speech interface.

3.2 Measures of Quality

Another important reason that people turn to advisory systems is to improve the *quality* of solutions to their task. This goal is especially common for problem-solving activities that involve many steps, like design or scheduling, but it is also relevant when one tries to find an appropriate item, like a book or Web page, from among many choices. As with efficiency, one can define the notion of quality in quite different ways.

If there exists some objective measure of quality for a domain, then one can use this directly as the dependent variable in an experimental study. For example, some popular advisory systems search the World Wide Web to find the site that offers a given item (a particular book or software package) at the lowest price. For such tasks, the resulting price constitutes an objective measure of the decision aid's success, which one can then compare to that for another advisory system or to the user's ability without computational support.

Evaluating quality becomes more complicated in domains that involve more than one criterion for success. For example, the Inca system described earlier operates in a domain where the user wants to minimize chemical spills, chemical fires, and hazard to human life. To evaluate the quality of system solutions, Iba et al. developed a simulator that could execute the generated schedules and then measured their percentage improvement on these dimensions over taking no action. But to obtain a single quality metric, they needed to combine these separate factors in some manner, and for simplicity they chose to give them equal weights. Table 1 (b) shows another result from their study, which suggested that Inca's seeding schedules with retrieved cases did not significantly improve the final quality measure over solutions produced from scratch or over seeding the repair process with schedules generated by heuristic search.

However, giving equal weight to different quality criteria conflicts directly with a core assumption of adaptive interfaces: that users differ in the relative importance they assign to such criteria. One obvious source for such information is the learned user model, but using this would be circular in that it would guarantee improvement in quality. In cases of multiple criteria, we need some external measure that is subjective but that is not tied directly to the user model, which may only partly reflect the user's true preferences.

3.3 Measures of User Satisfaction

These observations suggest reliance on some separate measure of user satisfaction to determine quality of the system's behavior. One way to collect this information would be to present each user with a questionnaire that asks about their subjective experience. Embedding a questionnaire in the system itself would make extra demands on the user, which seems undesirable, but that does not prevent a researcher from presenting a form to experimental subjects after they have finished using the adaptive interface. Yet questionnaires can be unreliable in predicting whether a person will continue to use the system, which we would like to know.

Another measure of user satisfaction involves giving the user some control over whether they use certain system features. If a user turns off the system's advisory capability or disables its personalization module after his initial interactions, we can safely conclude that he has not been satisfied by his experience with these features. Such subject control runs counter to normal experimental method, but one wants to know whether people will use an adaptive interface, so it seems appropriate. Some commercial advisory systems include disable switches, but we have not seen their use in any experimental studies of adaptive advisors.

3.4 Measures of Predictive Accuracy

Because the user model in an adaptive interface makes predictions about user responses to the system's advice, there is a natural temptation to rely on predictive accuracy as a surrogate measure for efficiency and quality. Moreover, accuracy is the most widespread measure in machine learning, which makes it very familiar to adaptive interface developers. Pazzani and Billsus provide one example in which they report the percentage of Web pages Syskill & Webert recommends that its users actually like. Gervasio, Iba, and Langley (1998) describe similar results with Inca in which they measured the percentage of user repairs to an initial schedule that the system correctly predicted. Both studies borrow directly from experiments on supervised machine learning, where the data sets include labels that the learned classifier must predict.

However, there are some inherent problems with using predictive accuracy to determine the success of an adaptive interface. Although this measure can be a useful analytical tool for understanding the details of system behavior, it does not directly reflect the overall efficiency or quality of solutions, which should be the main concern. Correct prediction of user responses may be correlated with these direct measures, but it cannot substitute for them. Also, some studies (including Gervasio et al.'s) involve collecting user traces in a nonadaptive setting, learning a user model from some of these data, and measuring the model's accuracy on the remainder. This scheme violates the standard assumption that adaptive interfaces change their user model over time, making the results of marginal relevance.

4. Independent Variables

A scientific experiment must do more than measure behavior under some condition. Because it aims to understand the factors that influence that behavior, it must measure the dependent variable in two or more situations that differ on some dimension. Because one can typically vary these factors independently, they are often referred to as *independent* variables. As with dependent measures, different controllable factors make sense for different disciplines. Here we consider four classes of independent variables that seem appropriate in the study of adaptive interfaces.

4.1 Effects of Experience

We have seen that adaptive interfaces infer user models by observing their user's behavior; this feature distinguishes them from traditional advisory systems, which remain static over time or which the user must reconfigure explicitly. However, their reliance on this approach makes it important that adaptive interfaces learn *rapidly*, since most users will want to see the feedback have an effect soon after they provide it. The issue here is not CPU time but rather the number of training cases before the system can accurately predict user preferences. Other things being equal, users will prefer adaptive interfaces that learn rapidly over ones that learn slowly. As Langley (1997) has noted, this concern with rapid learning encourages the use of simple induction algorithms, since they usually achieve reasonable accuracy before more sophisticated methods that have many more parameters.

This concern with learning rates also has implications for the evaluation of adaptive user interfaces. In particular, it suggests as a natural independent variable the number of training cases that the system has collected from the user. Plotting some performance measure against the number of training items produces a *learning curve*; such graphs are common in the psychological literature but remain rare in machine learning, where most researchers report results on training set of preselected size. In general, one hopes the learning curve for an adaptive interface will increase quickly in the early stages, even if the curve levels off as more data becomes available.

Figure 1 shows a learning curve from Rogers and Langley's studies of their Adaptive Route Advisor. Here the dependent variable is the percentage of route pairs for which the learned user model correctly predicts the route the subject prefers, averaged over 24 users and over ten training-test splits for each user. As expected, the accuracy increases quickly from around chance to 75% after 12 training pairs, then grows more slowly until it levels off at 79% at around 60 training

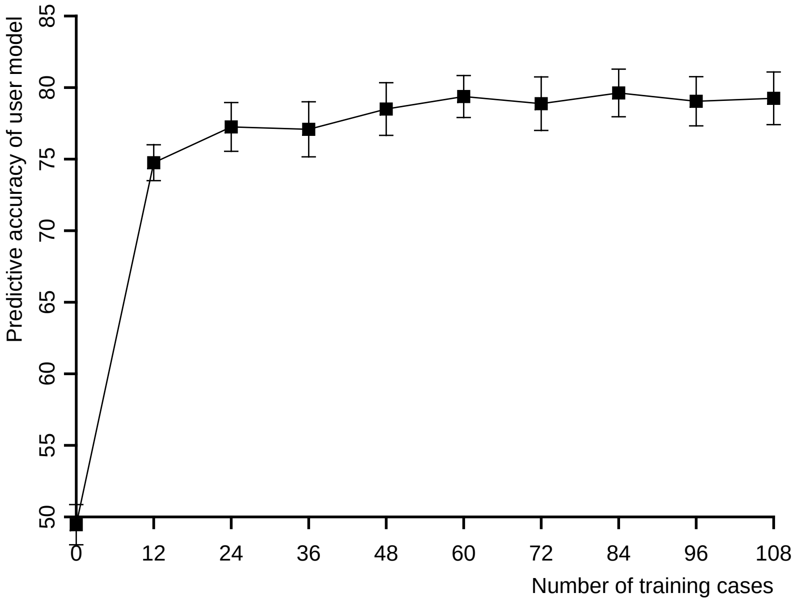

Figure 1. A learning curve that shows the percentage accuracy of a personalized user model generated by the Adaptive Route Advisor (Rogers & Langley, 1998), in predicting which of two routes a subject will prefer, as a function of the number of choices used in training.

pairs. Although more complex induction methods might have higher asymptotic accuracy after many more interactions, the Route Advisor's perceptron scheme serves it quite well in achieving reasonable accuracy quickly.

Pazzani and Billsus (1997), Hermens and Schlimmer (1994), and Gervasio et al. (1998) also report learning curves for their adaptive user interfaces, which suggests that they all recognize the importance of learning rate for their systems' success. However, most studies still collect user decisions in a non-adaptive setting, then use these traces to train and test the user modeling method off line. As we noted earlier, the data collected in such experiments can differ from those observed in actual system use, since adaptation can lead the advisory system to recommend different options as it updates its user model and since the users may react to these changes in system behavior.

4.2 Effects of the System

Another key claim of adaptive user interfaces, and computational decision aids in general, is that they help their users make decisions more effectively. Naturally, testing this claim requires dependent measures of effectiveness like those considered in the previous section. But it also requires a comparison between user behavior with and without the advisory system. Variations of this sort constitute an important independent factor in the experimental study of adaptive interfaces.

A clear advantage of adaptive user interfaces is that their interactive nature makes it easy to collect data on user behavior. But this also means that it is typically difficult to measure user performance in the absence of the interface. As a result, most experimental studies compare the full version of a system with a version that lacks certain features but that retains its interactive (often graphical) nature. Such *lesion* studies tell whether the omitted component actually aids user performance, but not whether users fare better with the lesioned interface than with no

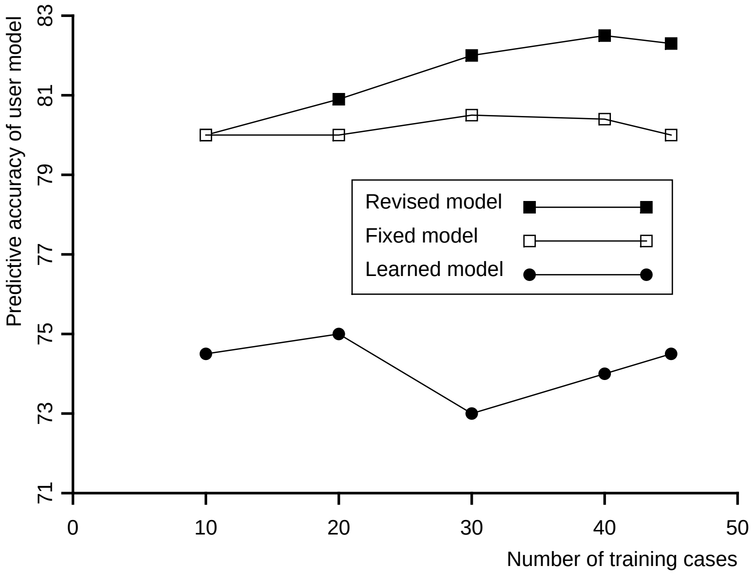

Figure 2. Results from a lesion study with SYSKILL & WEBERT that compared the strategy of using data to revise an initial user model to a version with a fixed user model and another that learned the user model only from data (Pazzani & Billsus, 1997).

computational aids at all. Most researchers simply assume the latter holds, although it could be tested empirically as well, with some difficulty.

Pazzani and Billsus (1997) report one lesion study with their SYSKILL & WEBERT recommendation system. They compared one version of their system, which based its user models on a combination of words that the user suggested and a set selected by cross validation, with a lesioned version that used only the former and a third that used only the latter. Figure 2 reproduces their learning curves from one domain, involving Web pages about biomedical topics, in which both lesioned systems did substantially worse than the full version of SYSKILL & WEBERT. Similar results occurred for two other domains, which led Pazzani and Billsus to conclude that both sets of features provided important sources of power for their recommendation system.

In some situations, it makes more sense to replace one component of the interface with another component than to remove it entirely. Such a *replacement* study contrasts system behavior using the standard module to behavior with another module that, intuitively, should not produce as good results. This straw man may use less information, use less computation, be less adaptive, or be otherwise more limited than its analog in the basic advisory system. The conclusions one draws from such experiments are the same as in lesion studies; if the straw man leads to worsened performance, then the standard module contributes to the success of the original system.

The form completer developed by Hermens and Schlimmer lends itself naturally to such a replacement study. Their experimental evaluation examined three conditions, one for the system's standard adaptation method, which relies on decision-tree induction, and two others for simpler induction methods: predicting the most recent value for a given field and predicting the most common value. These simpler techniques played the role of straw men, in that one would expect a system which relies on them to fare worse than one which relies on the decision-tree method, at least if the more sophisticated method is truly useful. The results of their study supported this conclusion, since the two straw men reduced keystrokes much less than the decision-tree module.

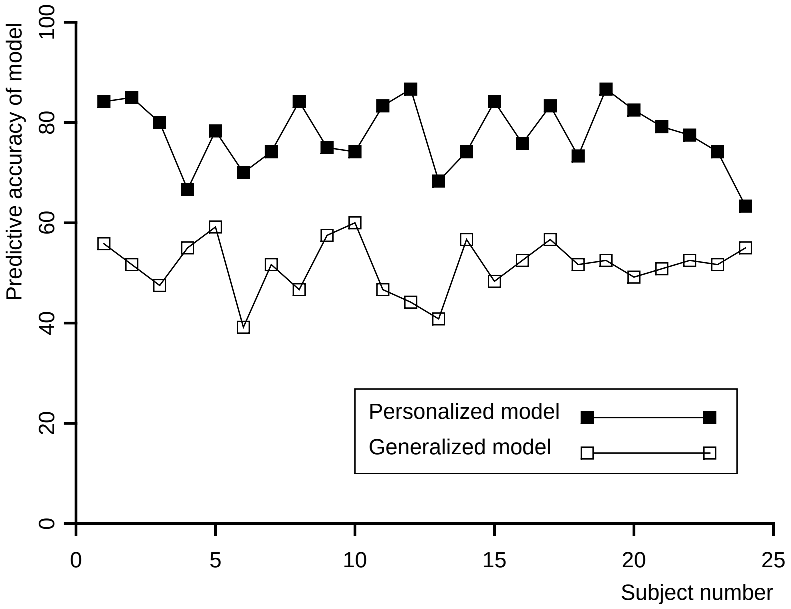

Figure 3. Percentage accuracy in predicting which of two routes a subject will prefer, with a personalized user model that was trained separately on each subject and a generalized model that was trained on choices from all subjects (Rogers & Langley, 1998).

4.3 Personalization and User Effects

Another type of independent variable that arises in the evaluation of adaptive user interfaces concerns the person using the system. The importance of user characteristics has long been recognized in human-computer interaction, where different types of interface may be appropriate for different types of users. Similarly, the notion of aptitude-treatment interaction has made its way from educational psychology into some computer-based tutors, which present material in different ways depending on student learning styles. Although such issues are relevant for adaptive user interfaces, they are less central than the claim that such systems benefit from adapting to *individual* users. One can best test such hypotheses about personalization by varying whether the system is tested on the same person as it was trained.

For example, Iba et al. (1998) report a personalization study with their interactive scheduler, INCA, that involved two separate users. In one condition, they used schedules from a given user's case library as the starting point for that user's repairs; this corresponds to the system's default mode. In the second condition, they presented each user with starting schedules from *another* user's case library. They predicted that subjects in the first setting would complete their revisions in less time, and produce schedules with higher quality, than those in the second situation, since schedules constructed by a particular user should reflect his preferences better than those created by another. However, their experiment revealed no significant differences between the two conditions on either dependent measure, suggesting that personalization at this stage of the system is less important than expected.

Rogers and Langley (1998) present a different approach to testing personalization claims in the context of their Adaptive Route Advisor. Their first experimental condition was analogous to that in the previous study, in that it tested a learned user model's ability to predict route preferences

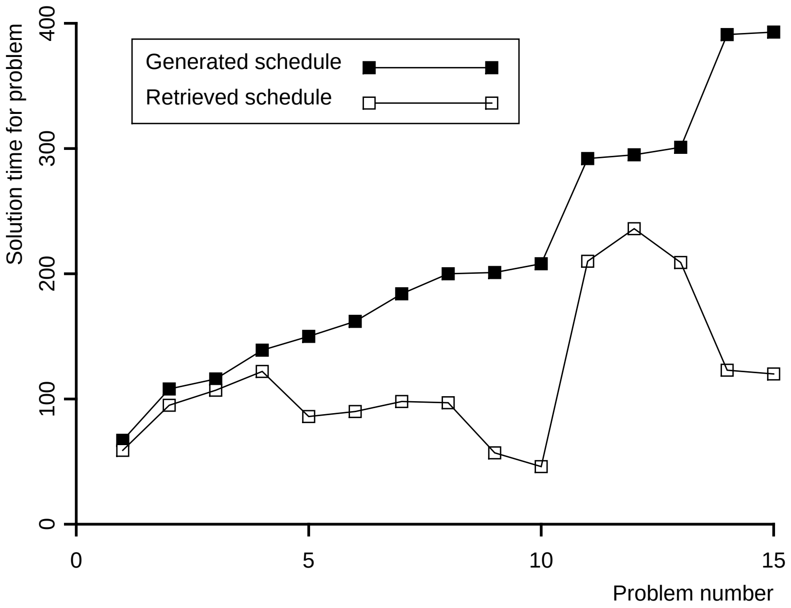

Figure 4. The time taken to repair a schedule as a function of task difficulty and whether the user of Inca (Iba et al., 1998) was initially given a retrieved or generated schedule.

for the user on which it was trained. But their second condition used not a model trained on a different user but rather a generalized model trained on decisions from 24 different subjects. Their hypothesis was that the personalized model would more accurately predict user responses than the generalized model, even though the latter had been trained on 24 times as much data. Figure 3 shows the results of this study, with accuracy graphed separately for each subject; in this case, the personalized models clearly fared better than the generic one.

4.4 Effects of Task Characteristics

A fourth important class of independent variables concerns characteristics of the task that the decision aid aims to support. In general, adaptive interfaces are intended to help users handle difficult tasks effectively, so most task variables involve some measure of problem difficulty. For instance, one can make a selection task more challenging by increasing the number of items available or increasing the number of features that describe each item. Similarly, one can make a configuration task harder by increasing the number of slots to be filled, the components possible for each slot, and the constraints that must be checked among them. The general prediction is that, as task difficulty increases, performance on the task decreases.

However, students of adaptive interfaces are less interested in task effects themselves than in the ability of their computational aids to minimize these effects. When present, this ability should appear as an *interaction* between task variables and system variables. That is, one expects that an increase in task difficulty will cause less reduction in performance when using an adaptive interface than when operating without such assistance. Thus, users will still make slower decisions and generate solutions with lower quality as they encounter more difficult problems, but the rate of reduction will be less than if the tackled the task without an advisory system.

An illustrative example of a task-oriented experiment comes from the Inca study (Iba et al., 1998) that compared the times taken to repair an initial schedule retrieved from a case library and another schedule generated by heuristic search. In a follow-up analysis, the researchers decided to order the scheduling tasks by their solution time under the second condition, which constitutes a rough measure of problem difficulty. Figure 4 presents the two resulting curves, which show that the time to repair generated schedules increases with problem difficulty, but that this trend is much weaker when users repair a schedule that Inca retrieves from its case library. Note that the definition of task difficulty here is somewhat circular, as it is linked to the dependent measure rather than being defined independently. Still, this experiment illustrates the interaction between task complexity and aspects of the advisory system.

5. The Context of Scientific Experimentation

The previous sections considered the types of variables that arise in experiments with adaptive user interfaces and some examples of these factors, but the larger context in which such experiments occur also deserves some discussion. The basic procedure differs little from that in other experimental sciences, except for the objects under study and their relevant characteristics. Thus, our points will appear obvious to some readers, but given the recent advent of adaptive interfaces and the paucity of studies, they are worth reiterating.

We should make clear at the outset a central assumption: the aim of systematic experimentation is not to show that one approach is superior to another but rather to increase *understanding*. Good science and good engineering both seek general principles that characterize the behavior of the objects they study, in this case adaptive user interfaces. Such understanding may lead eventually to improved artifacts, but the immediate goal is to produce results that add to our scientific knowledge.

5.1 Formulating Hypotheses

In many situations, a developer or researcher has clear expectations about the effects he will observe in an experiment. In such cases, he should state these hypotheses explicitly and use them to focus his experimental design. These predictions will often be qualitative in nature, such as the belief that system performance will improve with experience or that personalized models will fare better than generic ones, as illustrated by the examples in Section 4.

Nevertheless, they are enough to suggest clear dependent measures, likely independent variables, and the direction of the expected effect. Some studies are so exploratory in nature that no obvious hypotheses suggest themselves, but even brief reflection about the adaptive interface in question often brings plausible claims to mind. These predictions will often be violated in the experiment, but having stated them at the outset helps focus the scientist's attention and makes him prepared for such occurrences.

5.2 Designing Experiments and Selecting Samples

Having decided on a set of hypotheses, the researcher must design one or more experiments to test them. Since we have spent the preceding pages examining options for dependent and independent variables, we will not repeat them here. In most cases, the hypotheses themselves will suggest candidate variables, so the experimenter need only decide which measures best suit his purpose. But a complete design must also include decisions about the number of runs to average across in each condition, the range of each independent variable, and the step size to use for each such factor. For instance, to produce a learning curve, one must select the total number of training runs and the interval at which one will measure performance. If the independent variables are qualitative, one must specify the values to use. For example, in a replacement study, one must decide on the straw men that will serve in place of the system's normal module.

Another issue in experimental design involves sampling strategies. Because a researcher can never control all possible variables, he must collect multiple observations for each cell in the experimental design and average the resulting values. The fact that adaptive interfaces are artifacts lets one avoid some but not all of these complications. In particular, one has control over the user interface, the learning algorithm, and other system characteristics, but the users themselves bring with them all the complexities of experimental psychology. Thus, averaging over a sufficient number of subjects for each experimental condition is essential for a careful study of adaptive interfaces.

Basic experimental method recommends varying the value of one independent term while holding others constant. However, one can apply this process iteratively to obtain *factorial* designs, which observe the dependent measure under all combinations of independent values. This lets one move beyond isolated effects and look for interactions between independent variables. We have already discussed interactions that seem likely between task difficulty and computational advice, but one can easily imagine other interactions that could occur with an adaptive user interface. For instance, combining learning curves with a replacement study could reveal that one induction method achieves a reasonably accurate user model quickly but then levels off, whereas another method takes more training cases but has a higher asymptote. Interactions between user characteristics and induction methods also seem plausible in some domains.

5.3 Running Experiments and Compiling Results

Given a clear experimental design, a researcher can proceed to carry out the experiment that it specifies. To this end, he must recruit subjects, gather training and test problems, present subjects with the adaptive interface, have subjects use the system to solve problems or make decisions, and measure their performance on each task. The literature on experimental psychology incorporates many heuristics for the careful execution of experiments with human subjects, but we will not attempt to recount them here.

After measuring each subject's behavior for each time he has used the software, the experimenter can compile the results. This involves removing any suspicious measurements, averaging across the remainder for all subjects and problems in each experimental condition, and organizing the results in some readable format such as tables or graphs. Such statistics are the most obvious product of scientific experimentation, and we have seen many examples of experimental results in previous sections of the paper.

5.4 Testing Hypotheses

Once the experimenter has collected and organized his data, they can be used to draw tentative conclusions about the hypotheses that led to the study. In some cases, regularities in the data may suggest detailed models that would characterize them; for example, a learning curve may follow a power law that can be specified by its parameters. More often, one simply decides whether the data bear out the original qualitative hypothesis.

To this end, one can use statistical measures that indicate the confidence with which one can believe apparent differences. This confidence level is affected by three factors: the observed difference between conditions, the number of samples in each condition, and the variation in those samples. Even a large difference may not be robust if the sample is small or the variance high, making it desirable to use significance tests when possible. For presentation purposes, it is useful to report 'error bars' that show a confidence interval around the observed values, as we saw in Figure 1's learning curve.

5.5 Explaining Unexpected Results

Hypotheses about adaptive interfaces are based on some model of the system, the task environment, and its users, whether this model is explicit or not. Results that agree with an hypothesis lend evidence to that model, though they do not 'confirm' it; science can never draw final conclusions about any situation. Results that diverge from one's expectations count as evidence against a model, and thus require additional explanation. Accounting for a rejected hypothesis may involve altering assumptions about the users or the task, as we saw in the Inca study that found no advantages for personalization, or it may concern aspects of the system itself, such as the feedback mechanism or the induction algorithm.

In either case, faulty predictions indicate that one's model needs improvement, which often makes them more significant than positive results. More important, they can indicate directions in which to make changes. The ensuing altered models, whether formal or informal, suggest new hypotheses and predictions, which in turn suggest new experiments to test them. In other words, the iterative loop of hypothesize and test is as central to adaptive user interfaces as for any other experimental discipline.

6. Concluding Remarks

In this paper, we have reviewed the notion of adaptive user interfaces and examined some issues that arise in their experimental study. As an offspring of machine learning, cognitive psychology, and human-computer interaction, the field of adaptive interfaces has a rich conceptual and methodological inheritance to support its development. Yet the notion of software systems that adapt to their users introduces novel features that deserve careful reflection and innovative approaches to their empirical evaluation.

To this end, we reviewed four classes of dependent measures – efficiency, quality, user satisfaction, and accuracy – that seem appropriate for experiments with adaptive interfaces. We also examined four types of independent variables – amount of training data, system characteristics, user aspects, and task features – that seem likely to influence user-system behavior. We illustrated each type of variable with examples taken from previous studies of adaptive user interfaces, and we considered the broader context in which experimentation occurs. Many ideas carry over directly from other experimental sciences, but others seem unique to our growing discipline.

As the field of adaptive user interfaces matures, the need for such careful experimentation will expand rather than contract. This will result partly from the more complex artifacts made possible by early prototypes and their study. But it will also come from the very nature of science, where the process of discovery always leads to more questions than it answers, and where systematic experimentation occupies a central role in both the generation of questions and their resolution.

References

Boone, G. (1998). Concept features in Re:Agent, an intelligent email agent. *Proceedings of the Second International Conference on Autonomous Agents* (pp. 141–148). Minneapolis: ACM Press.

Gervasio, M., Iba, W., & Langley, P. (1998). Learning to predict user operations for adaptive scheduling. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 721–726). Madison, WI: AAAI Press.

Hermens, L. A., & Schlimmer, J. C. (1994). A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert*, *9*, 28–33.

Iba, W., Gervasio, M., Langley, P., & Sage, S. (1998). Experimental studies of intelligent assistance for crisis response. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society.* Madison, WI: Lawrence Erlbaum.

Lang, K. (1995). NEWSWEEDER: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 331–339). Lake Tahoe, CA: Morgan Kaufmann.

Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*, November, 55–64.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting Web sites. *Machine Learning*, *27*, 313–331.

Rogers, S., & Langley, P. (1998). Interactive refinement of route preferences for driving. *Proceedings of the AAAI Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems* (pp. 109–113). Stanford, CA: AAAI Press.

Schlimmer, J. C., & Hermens, L. A. (1993). Software agents: Completing patterns and constructing user interfaces. *Journal of Artificial Intelligence Research*, *1*, 61–89.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.