Users are individuals:- individualizing user models

ELAINE RICH

Department of Computer Sciences, The University of Texas at Austin, Austin, Texas 78712, *U.S.A.*

It has long been recognized that in order to build a good system in which a person and a machine cooperate to perform a task it is important to take into account some significant characteristics of people. These characteristics are used to build some kind of a "user model". Traditionally, the model that is built is a model of a canonical (or typical) user. But often individual users vary so much that a model of a canonical user is insufficient. Instead, models of individual users are necessary. This article presents some examples of situations in which individual user models are important. It also presents some techniques that make the construction and use of such models possible. These techniques all reflect a desire to place most of the burden of constructing the models on the system, rather than on the user. This leads to the development of models that are collections of good guesses about the user. Thus some kind of probabilistic reasoning is necessary. And as the models are being used to guide the underlying system, they must also be monitored and updated as suggested by the interactions between the user and the system. The performance of one system that uses some of these techniques is discussed.

1. Introduction

It has long been recognized that in order to build a good system in which a person and a machine cooperate to perform a task it is important to take into account some significant characteristics of people. The system can then be designed to take advantage of those characteristics, rather than to fight against them.

Traditionally, this has been done by collecting data on an average person's performance on various tasks in various environments. For example, Fitts' law (Fitts & Peterson, 1964) says that the time it takes for a person to move an object in his hand to a particular target position is proportional to log $2(2A/w)$ where A is the distance to be moved and w is the width of the target. This result suggests how the speed with which a person can operate a machine can be increased by increasing the size of the targets (such things as buttons and switches) that the operator must hit. As another example of this class of work, consider the large body of data on the relationship between the size of letters and their legibility (Smith, 1979). These results are important in the design not only of a wide variety of machines but also of other artifacts such as traffic signs.

The major weakness of these studies is that they make the assumption that the people who are involved constitute a homogeneous set. Under this assumption, the values that are determined to characterize a "typical" person can be used to design a system to be used by everyone. Although in most cases it is true that for at least the majority of the people, the system is better adapted to them than it would have been without those studies, it is not true that such a system is likely to be the best that could be produced. A much better system would be one in which the interface presented to each person was tailored to his own characteristics rather than to those of some abstract "typical" person. Although discussions of individual differences among users are rare in the human factors literature, they are not altogether absent. For example, Hudgens & Billingsley (1978) argue that sex is an important variable in human factors research. Another study, by Loo (1978), discusses individual differences in the perception of traffic signs. One reason that such studies have been rare is that it is often either too expensive or impossible to provide in physical devices the level of flexibility that they suggest. However, as we begin to see more and more of people's interactions with machines being mediated by computers under software control, it becomes possible to provide the flexibility necessary for truly personalized systems.

As a simple example, consider again the issue of letter size and legibility. If the letters of a display are being produced using a standard L.E.D. display, they will be the same size for all readers. But suppose the letters are being displayed on a CRT controlled by a computer. Now lines can be drawn wherever necessary to produce a wide variety of letter sizes as requested by individual users.

Recently, designers of user-computer interfaces have begun to focus attention on the needs of particular types of users. One group that has been frequently discussed is the class of "casual" users, who cannot be expected ever to use the system with a great deal of regularity [see, for example, Codd (1974) and Cuff (1980)]. This group must then be contrasted with the less well studied species, the regular, experienced user. Unfortunately, few systems will be

used exclusively by people of a single class. And it appears that system features that make life easy for one type of user make it correspondingly more difficult for another. For example, one study of the performance of expert users at a text editing task (Card, Moran & Newell, 1980) suggests that the number of keystrokes required to perform an operation should be minimized. Another study of people just learning to use an editor (Ledgard, Whiteside, Singer & Seymour, 1980) suggests that English-like, full word commands should be used. These conflicting requirements point to the need for a system that can appear differently to different users.

It is fortunate that the computer provides the means to increased personalization since it also produces a greater need for it by increasing the range of tasks for which people can hope to profit by dealing with machines. Tasks that were previously performed by people, such as collecting desired information from some kind of database, are now being done by computers. People who performed those tasks were able to accommodate the diverse needs of the other people with whom they dealt. For machines to assume those tasks and handle them satisfactorily, they too will have to be capable of accommodation to individual needs. In order to do that, they will have to exploit models of the individual users they encounter. This will require an expansion of the traditional notion of a "user model".

In the rest of this article, we will explore the issue of user modeling specifically in the context of computer software systems, both because of the increasing use of such systems by large groups of people and because of the inherent flexibility of such systems that makes effective modeling possible.

2. The space of user models

The term "user model" can be used to describe a wide variety of knowledge about people. The uses of user models span an equally-wide domain. The relationships among these diverse structures can be seen fairly easily if the universe of "user models" is characterized as a three-dimensional space. The dimensions, each of which will be discussed in more detail below, are:

1. one model of a single, canonical user vs a collection of models of individual users.
2. models specified explicitly either by the system designer or by the users themselves vs models inferred by the system on the basis of users' behavior.
3. models of fairly long-term user characteristics such as areas of interest or expertise vs models of relatively short-term user characteristics such as the problem the user is currently trying to solve.

There are other significant differences among the systems using these various types of user models, but they follow from these major differences. Systems with a single model of a canonical user can have that model permanently embedded within themselves, whereas systems with models of individual users must build the model on the fly, and so must make explicit the ways in which the model influences the performance of the overall system. Systems that extract the user model from the user's behavior must grapple seriously with the issues of incorrect or conflicting information arising from the inferences that led to the model. Systems with explicitly stated user information can, on the other hand, avoid many of those issues. Systems that deal with short-term knowledge must deal successfully with the problem of detecting when things change, while longer-term systems may be able to finesse that issue. But as these differences reduce to the three outlined above, they do not need to be focused on explicitly.

The next three sections briefly discuss how one might choose the best position in this three-dimensional space.

2.1. CANONICAL VS INDIVIDUAL MODELS

This dimension characterizes the major difference between "classical" human factors work and the more flexible models needed to provide the individualized interfaces that software control allows. A variety of computer systems have been designed around a canonical user model. For example, ZOG (Robertson, Newell & Ramakrishna, 1981) is a frame-based system that facilitates user-computer communication. Its design was heavily influenced by such factors as the response speed necessary to prevent user frustration. Another example of a system built around a model of a canonical user is Genesereth's automated consultant for MACSYMA, a symbolic mathematics package (Genesereth, 1978). The consultant exploits an explicit model of the problem solving strategy used by MACSYMA users. But, as suggested above, there is a limit to the usefulness of these canonical models to a system with a heterogeneous user community. Individual models can enable such systems to provide each user an interface more appropriate to his needs than could be provided using a canonical model. Of course, it is necessary to demonstrate that there exist techniques for implementing such models so that they actually do improve the performance of the system. A variety of such techniques will be presented in section 3.

The decision to exploit individual user models has a profound impact on the other aspects of user modeling. If a system possesses only a single model of a canonical user, that model can be designed once and then directly incorporated into the overall system structure. If, on the other hand, the system is ultimately to possess a large array of models corresponding to each of its users, the question of how and by whom those models are to be constructed arises. This leads to the second dimension in the space of user models.

2.2. EXPLICIT VS IMPLICIT MODELS

There are two ways to make systems different for different users. One is to allow users to modify the system to suit themselves. This is the approach taken by many systems that allow users to create explicitly their own environments within the system. Consider, for example, a computer program that enables system users to communicate with each other by sending mail messages back-and-forth. The program stores the messages in a set of files and provides functions by which users can read messages, answer them, and so forth. Such systems often allow users to set system parameters to determine such things as which message fields will be displayed when a message is printed. A much greater degree of personalization is provided by systems such as most implementations of the programming language LISP that allow users to specify an arbitrarily complex program that will automatically be executed whenever the user enters the system. With this facility, a user can create his own procedures, alter system variables, or define his own symbols. This same approach can be seen in many "personalized data base" systems (Mittman & Borman, 1975). In these systems, the personalization comes from the fact that each user can explicitly select documents and information of interest to him and store them in a private data base.

But this approach leaves quite a lot of responsibility in the hands of the user and is probably not appropriate for systems that expect truly naive users, i.e. people who will use the system only once or maybe two or three times, since a non-trivial amount of expertise is required in order to be able both to know what one wants to specify and how to specify it.

The other way to approach the problem of personalization is to provide the system with enough information about users that it can take charge of its own personalization. This can be done either trivially or in an intelligent way. A trivial example is a program that asks the user to rate his level of expertise with the system. The program then uses that level to determine how much information to provide in error messages. The program essentially contains models of how much information people at each level already possess.

A more sophisticated approach to automatic user modeling is needed in many systems in order to deal with both the need for more information about each user and the problem that users cannot always tell the system what it needs to know. Examples of this latter problem occur often in the domain of Computer Aided Instruction (CAI). A CAI system needs to know what each individual student knows, doesn't know, and knows incorrectly. The student, unfortunately, does not always know what he doesn't know, much less what he knows incorrectly.

Of course, students are not alone in their lack of knowledge about themselves. There is a lot of evidence in the psychological literature to support the assertion that people are not reliable sources of information about themselves [see, for example, Nisbett & Wilson (1977) and McGuire & Padawer-Singer (1976)]. In addition to the lack of accuracy inherent in explicit models, there is yet another consideration that argues for allowing the system to build its user models itself. People do not want to stop and answer a large number of questions before they can get on with whatever they are trying to use the system to do. This is particularly true of people who intend to use the system only a few times, and for only brief periods. To best serve these users, the system should form as good an initial model as it can and let the user immediately begin to use the system. This initial model can be based on the known characteristics of the system's overall user community, whatever additional information the system already has about each individual user (for example, his job title), and a set of facts characterizing a new user of the system. As the person interacts with the system, he provides it with additional information about himself. As it acquires this information, the system can gradually update its model of the user until eventually it comes to be a model of that individual as distinct from the canonical user. Using this approach, the greatest effort will be expended on the construction of models of frequent users, while much less effort will be expended on models of extremely infrequent users, models that would have little payoff in overall user satisfaction.

The most important implication of choosing to let the system build its own model of tJte user, based on the interactions between them is that most of the information contained in the model will be guesses. Thus the system must have some way of representing how sure it is of each fact, in addition to a way of resolving conflicts and updating the model as new information becomes available. Section 3 will suggest some ways of doing this.

2.3. LONG-TERM VS SHORT-TERM MODELS

In discussing the first two of these three dimensions, it was possible to argue that one form of user modeling would lead to a more habitable system than another. In discussing this third dimension, that is no longer the case. In order to interact reasonably with a user, a system must have access to a wide variety of information about him, ranging from relatively long-term facts, like his level of mathematical sophistication, to quite short-term facts, like the subject of the last sentence the user typed. Although all of this information can contribute to the habitability of a system, it is useful, at least at the beginning of an exploration into the topic of user modeling, to separate the problem of inferring long-term models from that of inferring short-term models because different techniques may

be appropriate for the solution of the two problems. .

It is probably reasonable to demand that the amount of effort spent to decide on a particular fact about a user be roughly proportional to the amount of time that fact will be able to be used. At one extreme, it is important that it not happen that so much time is spent trying to infer a fact that the fact is no longer relevant. At the other extreme, it may be reasonable to spend a lot of time, spread over many sessions, to form an accurate model of some essentially permanent characteristics.

There have been efforts devoted to both long-term and short-term individual user modeling. Short-term modeling is important in understanding natural language dialogue. Consider, for example, the following interchange:

Customer: How much is a ticket to New York?
Clerk: One hundred dollars.
Customer: When is the next plane?
Clerk: The next plane is completely booked, but there's still room on one that
                leaves at 8:04.
Customer: O.K., I'll take. it.

In order to make that response, the clerk had to refer to a model of the customer's current goal, getting to New York. It would have been inappropriate to have responded literally to the question and said simply 6:53. If computer systems are going to perform the task of the clerk in this example, then they too will need to be able to build and use models of the goals of their users. But models of such things as current goals are of fairly short-term use. The same customer could appear tomorrow intending to meet someone coming from New York and thus expecting a different response. Thus extremely responsive methods must be developed to perceive such goals and to notice when they change. For some more extended discussions of these sorts of issues, see Perrault, Allen & Cohen (1978) and Mann, Moore & Levin (1977).   .

But many systems could usefully exploit a large amount of much more stable knowledge about their users. These long-term models can be derived over the course of a series of interactions between the system and its users. The models can contain such information as the user's level of expertise with computer systems in general, his expertise with this system in particular, and his familiarity with the system's underlying task domain. In addition to these general things that could be of use in a wide variety of systems, the user models employed by a particular system will often need to contain specific information relevant to the system and its task domain. For example, in the librarian program to be discussed in section 4, each user model contains information about such things as a preference for books with fast-moving plots and a level of tolerance of descriptions of violence.

2.4. A REVIEW OF THE SPACE
Figure 1 shows the eight classes of user models generated by the three dichotomies we have just discussed, along with a few examples of each. The rest of this paper will focus on the lower right-hand corner.

	CANONICAL USER	INDIVIDUAL USER
EXPLICIT	It does not make sense for users to specify models of a canonical user, except possibly that users might describe themselves and the system might use many such descriptions to build a single model of a canonical user	SHORT-TERM If users specified short-term models, they would have time for little else LONG-TERM -Setting of parameters in systems such as a mail program -Systems, such as LISP, that allow arbitrary profile procedures to be run -Personalized data bases
IMPLICIT	SHORT-TERM Short-term data are so specific that they must be individual LONG-TERM -Game playing programs that assume opponent will play to win -ZOG -MACSYMA Advisor	SHORT-TERM - Conversational programs - CAI systems LONG-TERM -Grundy - Scribe helper

FIG. 1. The space of user models.

3. Some techniques for building user models

Having outlined both the need for user modeling and the kinds of approaches that can be taken toward it, some specific techniques that can be used for such modeling can now be presented. In this section, a variety of these techniques will be discussed:

- identification of the vocabulary and concepts employed by the user.
- gauging the responses with which the user seems satisfied.
- using stereotypes to generate many facts from a few.

These techniques fall into two broad groups: methods for inferring single facts at a time and methods for inferring whole clusters of facts at once. The next two sections discuss these two kinds of techniques.

3.1. INFERRING INDIVIDUAL FACTS

One of the simplest ways to derive information about a user is to look at the way he uses the system. Someone who begins a session with a series of advanced commands is probably an expert. Someone whose first few attempts to form commands are rejected by the system is probably a novice and needs some help. An easy way to implement user modeling based on this sort of information is to construct a dictionary of system commands, options, and so forth, and to associate with each item an indication of what information the use of that item provides about its user. This information can be on a variety of dimensions, such as expertise with the system and expertise with the underlying task. Once acquired, this information can be used to help decipher user errors and to produce messages with the right level of description.

Another way that users provide information about themselves is via the patterns of their commands. Suppose a

user asks a system for a particular piece of information. If he gets what he wants, he will either leave or get on with his next request. But if he does not get what he wants, he is likely to try to restructure his request in another attempt to get what he wanted. This attempt should signal to the system that it did not satisfy the user's need with its first response.

The use of both of these techniques can be illustrated by a brief examination of a system we are currently building, an interactive help facility for the document formatting system, Scribe (Reid, 1980). Users of this system can ask questions such as:

- How can I get an index generated?
- Why are the margins so wide?
- What is the difference between the itemize and the enumerate command?

The system stores its knowledge about Scribe as a set of condition-action rules. In their simplest form, these rules contain a single Scribe command as their condition, and the associated action describes the effect produced by the command. However, the way most commands operate is determined by the current values of some number of internal system variables, so these, too, must be mentioned as part of the rules' conditions. This often means that several rules, each with different conditions, all describe the operation of the same Scribe command. The description of the system's operation as given by these rules is hierarchical. The actions specified iri many of the rules are not primitive actions, such as place a character at a particular spot on a page, but rather are higher-level actions, often other Scribe commands, whose effects are in turn described by additional rules. The action component of many rules is the setting of some system variable that wi11later affect the operation of other rules.

This hierarchical organization of the information in the system makes it possible for the system to answer questions at many different levels of detail. So, for example, if the system is asked a "why" question such as "Why are the margins so wide?", it can respond either by stating the conditions that caused a particular rule to fire (e.g. "The value of lmarg is 10 and the value of rmarg is 10") or it can chain back through the rules to determine how those conditions could have come to be true. The appropriate level at which to answer a particular question is a function of the level of the question itself and the level of knowledge of the user who asked the question. In order to be able to decide on the correct level, the system maintains a dictionary that contains an entry for each of the things that can occur in the rules-both as actions and as conditions. Associated with each entry is information that describes when it may be appropriate to mention the associated concept in an explanation. For example, each concept has a rating that describes how expert a person would have to be at Scribe to be able to understand an explanation in terms of it. Thus many of the internal system variables have very high ratings, while the simple user commands have very low ones. Each concept also has a separate rating that describes the level of sophistication with computer systems necessary to understand it. For example, Scribe input files are block structures and Scribe's processing of them follows the standard block structure model. A programmer, even if he were a novice Scribe user, would understand an explanation in those terms, while a more Scribe-sophisticated typist might not. Whenever the help system tries to find an answer to a question, it first finds the rule (or rules) that apply to the particular situation. Then it looks at the concepts mentioned in those rules and compares what it knows about them to what it knows about the user who asked the question. If the levels match, a response is generated immediately. If they do not, the system chains through the rules, moving either up or down in the hierarchy as appropriate, until it finds an explanation at the correct level.

Of course, this method assumes that the help system has a model of the level of sophistication of the user. How can such a model be constructed? Fortunately, the same dictionary of concepts that was used when exploiting the model can also be used to construct it. When a user asks a question, he phrases it in terms of observed actions (such as where characters are on the page), Scribe commands, and Scribe parameters. The help system then matches this question to its rule base in its attempt to answer it. To do this, it looks up each of the elements of the question in its dictionary (which also serves as an index into the rules). People often refer to concepts simpler than the most complex ones they understand, but they do not talk about concepts more complex than the ones they understand. So the system can begin building its models of a new user by taking values associated with the first concepts he mentions. If more sophisticated concepts are mentioned later, the model of the user's level of expertise can be raised.

The model of the user can also be modified, as suggested above, by observing the patterns of the user's questions. Suppose the system misjudges the user's level and answers his first question by referring to a system parameter that means nothing to the user. The user's next question will almost certainly refer to that parameter in an attempt to figure out what it means. When the system sees this, it can conclude that its model is wrong and then modify it when it discovers the level of explanation with which the user is satisfied. Similarly, if the system underestimates

the user's knowledge, it will give him fairly broad, general answers, he will ask for more specific information, and the system can then update its model.

This kind of user modeling is very simple. It makes the, almost certainly unjustified, assumption that there is a fixed order in which people learn things about a system. Although this assumption is probably false, it is not completely wrong. The alternative approach would be to construct, for each user, a detailed model of exactly what he knows. This approach is necessary in CAI systems, which must control the teaching sessions with such models [see, for example, Self (1977)]. But this approach is very expensive, both in terms of the time it takes to construct the models and the space it takes to store them for a large number of users. The relationship between the user and the system is much looser in the context of a help system than it is in a CAI system. The user retains control of the interaction, and instead of being used in concentrated sessions to master ideas, help systems are normally used sporadically to solve particular problems. Thus the need for an exact model of the user's knowledge is less severe. Although there is, of course, no clear-cut line that can be drawn between these two types of systems, it does appear that in many situations, less than complete user models can be of use to a help system.

3.2. USING STEREOTYPES TO INFER MANY THINGS AT A TIME

The techniques that have been discussed so far enable a system to infer individual facts about a user. But if a user model is to be very complex, the question of how to collect all the required information within a reasonable period of time arises. Possibly a user will have only a few interchanges with a system, so user modeling that requires many interactions to build an initial model will be of little use. Fortunately, in many situations it is possible to observe one or a small number of facts and from them to infer, with a fair degree of accuracy, a set of additional facts. Human traits are not distributed completely at random throughout the population. Rather they often occur in clusters. These clusters can arise for a variety of reasons, such as the existence of a single factor that causes several traits to be present at once, or the existence of a causal chain among the traits themselves. For example, a person who is wealthy is likely to have travelled more than another person who is very poor.

People represent such knowledge about co-occurring traits in a collection of stereotypes. Although the word stereotype has. many negative associations, it is important to restrict its use here to the purely descriptive enumeration of a set of traits that often occur together. From this perspective, a stereotype is simply a way of capturing some of the structure that exists in the world around us. From the last couple of decades of work in artificial intelligence, we have come to understand the magnitude of the knowledge required to reason about the world. Fortunately, we have also discovered that that knowledge has a great deal of structure, which, if it can be captured, considerably constrains the things that must be considered at any one time. For example, events do not occur at random. Instead, common patterns of events, such as walking into a restaurant, getting a menu, ordering, eating, and paying, are observed. These event patterns have led to the development of scripts (Schank & Abelson, 1977), which have proved extremely useful in the construction of programs to understand descriptions of events such as those found in newspaper stories. Stereotypes provide a similar structure for information about people. Just as scripts are useful for doing the kind of reasoning about events required to understand newspaper stories, stereotypes are useful for doing the kind of reasoning about people required to build user models. In particular, they provide a way of forming plausible inferences about yet unseen things on the basis of the things that have been observed.

A stereotype represents a collection of traits. It can be represented as a collection of attribute-value pairs. We will call each such attribute a *facet*. A model of an individual user can also be represented as set of facets filled with values. The facets of the stereotypes used by a system should correspond to the facets of the user models built by the system. For example, one of the traits it might be useful to consider is the user's level of experience with a particular system. So models of individual users as well as the appropriate stereotypes would contain the facet "expertise", which could take on values, say, from 1 to 10.

Some traits may be easily observable. They serve as *triggers* that cause the activation of the entire stereotype. Since the presence of a trait may only be suggestive of a particular stereotype rather than absolute evidence for it, each trigger has associated with it a rating that is a rough measure of the probability that the stereotype is appropriate given that the trigger was observed. Of course, it is not only the relationship between triggers and stereotypes that is at best suggestive. A stereotype says only that a collection of traits often occur together, not that they always do. So, associated with each facet in the stereotype must be a rating that estimates the probability of the corresponding trait given the appropriateness of the stereotype.

Stereotypes represent structure among traits. There is often additional structure that can be captured by representing a collection of stereotypes as a hierarchy. Information in very general stereotypes can be used unless conflicting information is suggested by more specific stereotypes. The most general stereotype available to a system

can represent a model of a canonical user. Thus even without much information, a system built on stereotypes will do no worse than one built on the traditional built-in model of a canonical user.

One of the most important problems to be addressed in any user modeling system based on inferences from the user's behavior is how to detect and resolve conflicts between inferences. In order to facilitate this, ratings are attached both to triggers and to each prediction (facet) of each stereotype. In addition, each facet of an individual user model must contain not just a value, but also an estimate of the system's confidence in that value (which can be used to determine how much the value should be allowed to influence the performance of the overall system) and a list of reasons why the value is believed. This list of reasons is important. For example, suppose a stereotype isactivated as a result of some observed trait of a user. That stereotype predicts a value for a particular facet, but the system's model of the user already contains a different value for that facet. If the system remembered where it got that value from, it may be possible to resolve the conflict fairly easily, as, for example, if the earlier value came from a stereotype more general than the one just being activated.

Sometimes, several different stereotypes may predict the same value rather than different ones for a facet. In this case, the system's level of confidence in the prediction can be higher than it would be if only one source of information were present. Since it is reasonable for stereotypes to predict other stereotypes and for particular facet-value pairs to predict stereotypes, the influx of new information may necessitate the propagation of rating changes throughout the user model. The exact extent to which it is effective to continue such propagation is an issue that needs to be determined empirically.

All of these methods of combining the inferences suggested by various stereotypes can be generalized to form the basis for integrating a wider variety of sources of knowledge about an individual user. Helpful as stereotypes can be at enabling the initial construction of a user model fairly quickly, they do not eliminate the need for other kinds of information, including both the answers to direct questions and the other indirect techniques discussed above in the context of the Scribe helper. But once each element of the user model is accompanied by a rating and a list of the evidence supporting it, it is easy to add arbitrary sources of knowledge. New information that supports old values causes the ratings attached to the values to increase. New information that conflicts with old values causes the conflict resolver to examine the credibility of the competing voices to produce as good an estimate of the truth as possible. The user's answers to direct questions can be given priority over simple inferences, which can in turn be given priority over the predictions of stereotypes.

The discussion so far of the use of stereotypes has been general and has avoided reference to particular systems or task domains. In the next section, a specific system that used stereotypes successfully to build models of its users will be discussed. The general points mentioned in this section will be illustrated with concrete examples.

4. Grundy: a case study in the use of stereotypes

In order to test many of the ideas outlined above, a pilot system called Grundy was built. Grundy recommends novels that people might like to read. To do this, it exploits two collections of data:
- descriptions of individual books. Each description is a set of facets filled with appropriate values.
- stereotypes that contain facets that relate to people's taste in books. Associated with each stereotype is a collection of triggers.

In addition, Grundy has some knowledge about each of the facets that can occur in the stereotypes. This knowledge is used to help to resolve conflicts between competing inferences and also to map from information in the user model to information in the book descriptions. For a description of Grundy more complete than the one presented here, see Rich *(1979a, b)*.

When a new user begins a conversation with Grundy, he or she is asked to provide a few words that (s)he thinks provide a good self-description. Grundy uses those words as triggers for the appropriate stereotypes, and it begins building its model of the user. Usually several stereotypes are activated at this point and often there are conflicts among their predictions, which Grundy resolves as well as it can. Grundy then estimates (using a combination of the number of things it believes and how strongly it believes them) whether it has enough information to begin recommending books. If it does, then it goes ahead. If it does not, it asks the user for a few more words.

Figure 2 shows two of the stereotypes used by Grundy. Figure 3 shows what Grundy's model of a user would look like after being told the user's name (and deducing from it that she is female) and after specifically activating the WOMAN, FEMINIST, and INTELLECTUAL stereotypes. Whenever Grundy activates a stereotype it also activates all of that stereotype's generalizations, so the stereotypes EDUCATEDPERSON and ANY-PERSON (the canonical user model) have also been activated.   Notice that each of the stereotypes contains only some of the facets contained in a complete user model. This often occurs since many stereotypes involve only one (or perhaps a few) significant aspects of a person.

FACET	VALUE	RATING
Activated-by	Athletic-w-trig	
Genl	ANY-PERSON	
Movitivations		
Excite	800	600
Interests		
Sports	900	800
Thrill	5	700
Tolerate-violence	4	600
Romance	-5	500
Education	-2	500
Tolerate-suffering	4	600
Strengths		
Physical-strength	900	900
Perseverance	800	600
	SPORTS-PERSON	

FACET	VALUE	RATING
Activated-by	Feminist-w-trig	
Genl	ANY –PERSON	
Genres		
Women	800	700
Politics	Liberal	700
Sex -open	5	900
Piety	-5	800
Political-causes		
Women	1000	1000
Conflicts		
Sex-roles	900	900
Upbringing	800	800
Tolerate-sex	5	700
Strengths		
Perseverance	800	600
Independence	800	. 600
Triggers	Fem-woman-trig	
	FEMINIST	

FIG. 2. Some sample Grundy stereotypes.

After it accumulates enough information to get started, Grundy begins recommending books one at a time until the user tells it to stop. The process of choosing a book proceeds as follows.

1. Select a salient facet in the user model. Salient facets are those with non-middle-of-the-road values and high ratings.
2. Use an inverted index into the book data base to select all the books suggested by this particular facet value.
3. Compare each of the chosen books to the user model along all dimensions. Eliminate books that exceed certain thresholds (such as tolerance for violence).
4. Of the books that have not been eliminated, choose the one that is the best match. If it is above a threshold of closeness of match, recommend it. Otherwise, go to step 1, choose a new facet, and try again.

Gender	female	1000	Inference-female name
			WOMAN
Nationality	USA	100	ANY-PERSON
Education	5	900	INTELLECTUAL
Seriousness	5	800	INTELLECTUAL
Piety	-3	423	WOMAN
			FEMINIST
			INTELLECTUAL
Politics	Liberal	910	FEMINIST
			INTELLECTUAL
Tolerate-sex	5	700	FEMINIST
Tolerate-violence	-5	597	WOMAN
Tolerate-suffering	-5	597	WOMAN
Sex-open	5	960	FEMINIST
			INTELLECTUAL
Personalities	4	646	WOMAN
Opt-pes	0	100	ANY-PERSON
Plot-intr	0	100	ANY -PERSON
Plot-speed	-2	475	EDUCATED-PERSON
Suspense	0	100	ANY-PERSON
Thrill	-4	839	WOMAN
			INTELLECTUAL
Romance	3	696	WOMAN
Confusion	3	570	EDUCATED-PERSON
Real-fant	0	100	ANY -PERSON
Clmedy	0	100	ANY-PERSON
Genres			
Literature	900	700	INTELLECTUAL
Women	800	700	FEMINIST
Political-causes			
Women	1000	1000	FEMINIST
Strengths			
Perceptiveness	700	570	EDUCATED-PERSON
Intelligence	900	800	INTELLECTUAL
Independence	800	600	FEMINIST
Perseverance	800	600	FEMINIST
Sympathy	700	497	WOMAN
Kindness	700	497	WOMAN
Weaknesses			
Reason	800	600	INTELLECTUAL
Conflicts			
Difference	800	600	INTELLECTUAL
Upbringing	800	800	FEMINIST
Sex-roles	900	900	FEMINIST
Propriety	700	497	WOMAN
Love	700	497	WOMAN
Motivations			
Learn	900	700	INTELLECTUAL
Interests			
Ideas	900	900	INTELLECTUAL

FIG. 3. A sample Grundy user model.

Having selected a book, Grundy tells the user its author's name and the title and then asks her whether she has read it before. If she has, Grundy knows that it is on the right track. It can now reinforce its belief in the things that led it to choose this book. (See section 5 for more discussion of the issue of modifying Grundy's data base.) If she

did not like the book, Grundy needs to find out why. Ideally, it would simply say, "Why not?", but far more knowledge than Grundy has would be required to interpret answers to such a question. For example, someone might say that she did not like the book because the main character reminded her of her dentist. So instead, Grundy tries to find out which of the beliefs that it has about the user and that it used to choose that book was wrong. To do so, it asks a few direct questions until it either locates the problem or is forced to give up. If it found the problem, then it can update both its model of the user and its database of stereotypes.

If the user tells Grundy that she has not read the book, then Grundy tells her some things it thinks would interest her about it. Grundy uses its model of the user to choose which of the book's characteristics to mention. Then it asks the user whether she thinks she would like the book. This time, if she says yes, Grundy does nothing, since the positive response is based only on the few facts the user has seen. But if the user says the book does not look interesting, Grundy uses its procedure described above to try to find out what went wrong, so that it can try to find something she will like better.

In order to test the usefulness of Grundy's user models, an experiment was conducted in which Grundy gave each of its users as many suggestions as they wanted. It also gave them each several suggestions chosen at random without the aid of the user model and asked them whether the suggestions looked good. These served as a control. Table 1 shows the percentage of suggestions that were described as good as opposed to those described as bad, both in controlled mode (where the user model determined the selection) and in random mode. These numbers show that Grundy does significantly $(p < 10\text{-}9)$ better with the user model than without it.

Although Grundy's models of its users do not come close to capturing ali of the intricately connected factors that determine which novels a person will like, its rate of success at making suggestions indicates that less than complete user models can provide useful guidance to interactive systems.

Table 1		
The usefulness of Grundy's user models		
	Controlled	Random
Good	72%	47%
Bad	28%	53%

5. Learning in Grundy

Stereotypes are extremely useful in enabling a system to build quickly an initial model of a new user so that it can get on with whatever the real task is. But how can we develop accurate stereotypes for a system to use? Will all of the user modeling efforts be futile if the stereotypes are inaccurate? These are important questions that still need complete answers, but experience with Grundy suggests that they do not represent insurmountable obstacles. Grundy's initial stereotypes represented merely my intuitions about people and the books they read. No attempt was made to collect any hard data. Despite this, the stereotypes are very useful. But the more interesting thing that can be observed is that Grundy is able to modify its stereotypes on the basis of its experience when that experience contradicts the predictions of the stereotypes. Only fairly simple modifications can be made. Facets cannot be added or deleted, but the value of a facet can change, as can its rating.

Grundy's initial stereotype for a typical male reader indicated that men like to read books with fast-moving plots and a lot of thrill and excitement. This may be true of the male population of the U.S.A. But the men Grundy actually saw were not a very broad cross-section of that population; they were all university faculty and graduate students. So they tended to like intellectual sorts of books, which tend to be stronger on philosophy than on plot. So Grundy gradually modified its MAN stereotype to reflect the tastes of the population of men it actually saw rather than some population I had imagined. The fact that it could do this suggests two encouraging things about the merits of the use of stereotypes in user modeling systems:

- it is not critical that exact data describing the user community be available in order to build an initial set of stereotypes.
- if a single system is used in a variety of communities its stereotypes can evolve separately in each of them so that each is characterized fairly accurately.

Other types of learning that Grundy does not do are, of course, possible. Grundystores all of its user models so that when a user returns for later sessions, a new model need not be built from scratch. Thus it would be possible to construct entirely new stereotypes by observing patterns of traits that occur commonly among the users.

6. Conclusion

In this paper, I have argued that for many interactive computer systems, the user community is sufficiently heterogeneous that a single model of a canonical t1ser is inadequate. Instead, the ability to form individual models of individual users is needed. And then I have shown that besides being necessary, such models are also possible, and a collection of ways in which they can be built and exploited has been presented.

The thing all of these techniques have in common is that they involve guesses about the user. These guesses are made by the system on the basis of its interaction with the user. As a result, the possibility of error must always be considered. To handle this, the system must do two things:

- it must attach ratings and justifications to each of the things it believes.
- it must not regard the user model as fixed, but rather as something upon which it can continuously improve by collecting feedback from the user on each interaction.

References

CARD, S. K., MORAN, T. P. & NEWELL, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the Association for Computing Machinery, 23,* 396-410.

CODD, E. F. (1974). Seven steps to rendezvous with the casual user. In KLIMBIE, J. W. & KOFFEMAN, K. L., Eds, *Data Base Management.* Amsterdam: North-Holland.

CUFF, R N. (1980). On casual users. *International Journal of Man-Machine Studies, 12,* 163-187.

FITTS, P. M. & PETERSON, J. R (1964). Information capacity of discrete motor responses. *Journal of Experimental Psychology,* 67, 103-112.

GENESERETH, M. (1978). An automated user consultant for MACSYMA. *Ph.D. thesis,* Harvard University.

HUDGENS, G. A. & BILLINGSLEY, P. A. (1978). Sex: the missing variable in human factors research. *Human Factors,* 20, 245-250.

LEDGARD, H., WHITESIDE, J. A., SINGER, A. & SEYMOUR, W. (1980). The natural language of interactive systems. *Communications of the Association for Computing Machinery, 23,* 556-563.

LOO, R (1978). Individual differences and the perception of traffic signs. *Human Factors, 20,* 65-74.

MANN, W. C., MOORE, J. A. & LEVIN, J. A. (1977). A comprehension model for human dialogue. In *Proceedings International Joint Conference Artificial Intelligence,* 5, 77-87.

MCGUIRE, W. J. & PADAWER-SINGER, A. (1976). Trait salience in the spontaneous self-concept. *Journal of Personality and Social Psychology,* 33, 743-754.

MITTMAN, B. & BORMAN, L. (1975). *Personalized Data Base Systems.* Los Angeles: Melville Publishing Co.

NISBETT, R E. & WILSON, T. D. (1977). Telling more than we can know: verbal reports on mental processes. *Psychological Review,* 84, 231-259.

PERRAULT, C. R, ALLEN, J. F. & COHEN, P. R (1978). Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the Second Conference on Theoretical Issues in Natural Language Processing.*

REID, B. K. (1980). Scribe: a document specification language and its compiler. *Ph.D. thesis,* Carnegie-Mellon University.

RICH, E. A. *(1979a).* Building and exploiting user models. *'Ph.D. thesis,* Carnegie-Mellon University.

RICH, E. A. *(1979b).* User modeling via stereotypes. *Cognitive Science,* 3, 329-354.

ROBERTSON, G., NEWELL, A. & RAMAKRISHNA, K. (1981). The ZOG approach to man-machine communication. *Internal Journal of Man-Machine Studies,* 14 (4), 461-488.

SCHANK, R C. & ABELSbN, R P. (1977). *Goals, Plans, Scripts and Understanding: An Enquiry into Human Knowledge Structures.* Hillsdale, New Jersey: Erlbaum Press.

SELF, J. A. (1977). Concept teaching. *Artificial Intelligence,* 9, 197-221.

SMITH, S. (1979). Letter size and legibility. *Human Factors,* 21, 661-670.