Naval Research Laboratory
Washington, DC 20375-5000

NRL Report 9148

Adaptive Human-Computer Interfaces

A. F. NORCIO AND J. STANLEY

Human Computer Interaction Laboratory
Information Technology Division

September 30, 1988

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Report 9148	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b. OFFICE SYMBOL (If applicable) Code 5533	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61153N		RR015-09-42	55-3106-0-8

11. TITLE (Include Security Classification)
Adaptive Human-Computer Interfaces

12. PERSONAL AUTHOR(S)
Norcio, A. F. and Stanley, J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 1/1/88 TO 4/30/88	1988 September 30	20

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Adaptive systems
			Human-computer interface
			Software design

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Interface software that can adapt to the current user and the current context is a long-term research goal of the Adaptive Interface project at the Naval Research Laboratory's Human-Computer Interaction Laboratory. This report presents a survey of recent research in adaptive interface computer software as well as a discussion of factors that require consideration in designing this software. An adaptive interface needs to include a knowledge base that encompasses four domains. These four domains are knowledge of the current user, knowledge of the interaction scheme, knowledge of the problem task, and knowledge of the underlying system. This report reviews and discusses these knowledge bases along with the positive and negative aspects of adaptive interfaces.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Anthony F. Norcio	22b. TELEPHONE (Include Area Code) (202) 767-3249	22c. OFFICE SYMBOL Code 5533

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

INTRODUCTION 1

BACKGROUND 2

Negative Aspects of Adaptive Human-Computer Interfaces 3
Positive Aspects of Adaptive Human-Computer Interfaces 3
Knowledge Required by an Adaptive Interface 4

KNOWLEDGE OF THE USER 4

Modeling the User 4
Constructing User Models 6

KNOWLEDGE OF THE INTERACTION 8

Natural Language Interfaces 8

KNOWLEDGE OF THE TASK/DOMAIN 10

Task Modeling 10
Goal Detection and Plan Inference 10
Help Systems 11

KNOWLEDGE OF THE SYSTEM 12

Input/Output Issues 12

CONCLUSIONS 13

ACKNOWLEDGMENTS 13

REFERENCES 14

ADAPTIVE HUMAN-COMPUTER INTERFACES

INTRODUCTION

In many computer applications people frequently are presented with exhaustive amounts of data and must make critical decisions in brief time intervals. Such situations are often made worse since the users lack the knowledge and experience to be effective. Because of this and because hardware and applications are becoming more complex, the area of human-computer interface (HCI) technology is becoming essential to the operational success of many systems. Unfortunately, the technology to support the development and implementation of effective user computer interfaces is lacking. At least one authority warns, "unless we pay close attention to the user interface, users will become hopelessly lost and ineffective ... the complexity of the systems should be transparent to users" [1]. To underscore the problem further, Defense Advanced Research Projects Agency (DARPA) officials now contend that interface research is lagging while other fifth-generation research efforts are progressing [2]. Some HCI guidelines exist [3, 4], but they are not designed to provide the necessary information for determining the effectiveness and appropriateness of specific interfaces.

At least three major factors underly the inadequacy of HCI technology. The first is that interface software is generally not viewed as part of the system but rather as a software package between the system and the user [5]. This traditional approach keeps both the interface and the user external to the system rather than part of it. The result is a fragmented operation in which an interface is frequently not well suited to the system or to the user, and more often to neither.

The second factor contributing to the inadequacy of HCI technology is that the design of effective interfaces is a difficult problem with sparse theoretical foundations [6]. Though components of a theoretical framework have been suggested [7], extensive experimental investigations are needed before a coherent theory can be advanced. Without theoretical models on which to base HCI design principles, user effectiveness can only be at best moderately enhanced. In this instance, the old adage that nothing is more practical than a good theory is appropriate. We must be careful not to view technology enhancement as technology advancement. It is difficult to imagine significant HCI technology advancements until a sound theoretical foundation is established.

The third and final factor hindering HCI progress is that software engineering principles are generally not given significant consideration in designing interfaces. Specifically, user specifications using the information-hiding principle [8] in an abstract interface [9] need to be incorporated in the design of human-computer interaction software.

Some of these concerns are currently being addressed with user interface management systems (UIMS) [10]. The thrust of the UIMS approach is to make HCIs a separate and important software design concern to which software engineering techniques are applied. However, cognitive models that relate to human performance are rarely considered in this approach. The work needs to be extended by incorporating cognitive models of users that are abstract specifications of user populations.

Manuscript approved June 10, 1988.

The concept of an adaptive interface [11, 12] is an extension of UIMS. The idea of an adaptive interface is straightforward; simply, it means that the interface should adapt to the user rather than the user adapt to the system. In spite of this apparent and simple fact, the implicit problems in adaptive interfaces are fundamentally difficult and complex [13]. One of the first issues that needs to be addressed is the problem of user models [14, 15]. Recent studies have shown that user models that underlie adaptive systems must be based on theories of cognition and must explain evolving changes in user performance and capability [16, 17]. The purpose of the models is to determine users' levels of expertise and experience by collecting input parameters such as command types, error rates, and speed [18, 19].

Another critical aspect of an adaptive interface is the dialogue between the user and the system [13]. The dialogue must be appropriate for the specific user [20, 21]. Also, it is suggested that the application plays a major role in the success of adaptive interfaces [22]. It appears, however, that no study has attempted to examine these interrelated issues in the context of a unified approach.

The structure and architecture of the interface [23] is equally important. An adaptive interface must be an integral component of the overall system so that the adaptation can take place in the context of the application. If this is to be accomplished, software engineering techniques (for example Ref. 24) need to be explored for designing adaptive interfaces.

BACKGROUND

The functionality of the system is usually the main concern in system design. The user interface, which is the component of the system that communicates with the user, is typically considered the incidental part of the system and frequently is viewed as an afterthought. However, the importance of the interface is currently gaining more attention. This is evident in the new and extremely effective interface designs that have surfaced. These include the use of different input/output devices such as the mouse, the light pen, the touch screen as well as the innovative presentation methods such as windows and icons.

These developments make computer systems easier to use, and a larger variety of users avail themselves of them. The idea that the interface is an integral part of the system and not merely gateway to the system is now widely accepted. However, the computer is still not the completely supportive tool that it potentially could be. Individual users differ on various dimensions. On most systems, the users must adjust their behavior and problem solving strategies to the system. That is, the system is designed for the average user, but not for all users. Any person who interacts with the system must adapt to the system. An ideal computer system should adapt to the current user by compensating for weaknesses, by providing help appropriate to the context, and by decreasing the mental and physical workload of the particular user.

An interface that can be adapted to the user would be more complex than one that cannot. A system can be made adaptive in two ways. The first way is to allow the user to make modifications if the behavior of the system is unsatisfactory once it is in operation. Edmonds [11] discusses interfaces that may be modified by several classes of people. A computer specialist, a trained user, or any user may modify the interface. The amount of change that is allowed depends on the user who is making the modifications and on the access privileges that are allowed to the internals of the interface. Although this may produce a better interface, it leaves the burden of adapting to the user.

The second form of adaptation is dynamic adaptation by the system itself. In this report, an interface that dynamically modifies itself is what is meant by the term *adaptive interface*. An adaptive interface needs information that generally is not required or available to a static interface. Recently, research interest has increased in the area of system adaptation through machine learning.

Any form of machine learning can be called adaptation since the machine assimilates new information and responds more appropriately to new situations. However, machine learning does not always constitute an adaptive interface. The adaptive interface changes with respect to the particular user and current context while a machine that can *learn* may behave the same way with all users. Thus, an adaptive system becomes an adaptive interface only when it learns with respect to the individual user and not when it learns only with respect to the task domain. That is, an adaptive interface works differently depending on the current context. This includes both the current task and the current user.

Negative Aspects of Adaptive Human-Computer Interfaces

The concept of an adaptive interface is often criticized. Greenberg and Witten suggest some reasons why this may not be desirable [25]. First, the user may not be able to develop a coherent model of the system if the system changes frequently. This may undermine the user's confidence and performance with the system. If the user does not have a clear understanding of the system behavior, the user's effectiveness can be seriously reduced.

Another problem that may arise with an adaptive system is the loss of control or the feeling of loss of control that the user may experience. Wahlster and Kobsa note that users may attempt to disguise their goals and preferences [21]. They suggest that the interface should allow the users to inspect a comprehensible version of their models. Also, the system should allow the users the option of *turning off* the modeling component of the interface. However, an adaptive interface is not designed to take control from the user, but rather, it is intended to provide the maximum and most appropriate assistance to a given user for the current task.

Another disadvantage of adaptive interfaces is an increase in implementation complexities and costs. Although this may be a problem now, as development of adaptive interfaces continues, the cost of implementation should decrease. Also, adaptive interfaces incur a higher computational overhead. Any interface that has a complex modeling component must do more computation than a system that does no modeling. This uses computational resources and consequently may increase the system response time. But, the advances in hardware technology result in lower costs and faster systems.

Positive Aspects of Adaptive Human-Computer Interfaces

However, adaptive interfaces have several positive aspects. System automation is an area that lends itself well to the need for an adaptive interface. A system that dynamically allocates tasks must be able to adapt to individual users. Rouse suggests that tasks should be allocated to both the user and the computer [26]. He further suggests that this allocation should depend on whether the user or the system has the better resources for performing the given task. This depends on the specific situation, the individual user, and system capabilities. Consequently, it is imperative to have information specific to the current human operator for an optimal allocation process. This is because the amount and type of information people can process as well as the way in which they process it varies.

For example, when human operators are confronted with an overwhelming amount of information in a decision-making task, each must decide what information to request and use. In such environments, it has been discovered that human operators usually do not produce optimal behavior compared to the Bayesian normative model of the task [27]. However, all operators do not differ in the same way or direction from optimal behavior. Each individual operator exhibits consistent biases in one direction or the other. For optimal overall system performance, the computer should be able to compensate for the inherent biases of the operator.

The computer must also adapt for other existing reasons. Many times users do not have the necessary information or expertise to modify their behavior. Indeed, they may not know that modifications can or need to made. Because of the increase in the availability of computers, the number of novices who use computer systems is increasing dramatically. Adaptation is particularly useful for novices. Also, with an adaptive system the user's proficiency with a new system is increased and frustration with an overly simple system is prevented [25].

The issue is whether the advantages of adaptive interfaces outweigh the costs of constructing and executing them. However, the main criticism of adaptive interfaces, cited above, can be overcome without eliminating the adaptive mechanisms. If they are carefully designed, the adaptive interface makes the system more useful to a larger number of people. Novices and experts can use an adaptive system with equal ease. They also enable particular users to use the system more efficiently by providing them with the proper kind and amount of assistance for their individual needs.

Knowledge Required by an Adaptive Interface

An adaptive interface needs to include a knowledge base that encompasses four domains. These are [22,13]:

- knowledge of the user—that is, expertise with the system;
- knowledge of the interaction—that is, modalities of interaction and dialogue management;
- knowledge of the task/domain—that is, the ultimate purpose of the problem area and its goals;
- knowledge of the system—that is, the system's characteristics.

Each of these domains is discussed in detail throughout the remainder of this report.

KNOWLEDGE OF THE USER

An adaptive system must be able to characterize and distinguish between individuals. A user model that combines information about the user's knowledge, capabilities, and preferences should be constructed for each user. This model should reflect the content of the user's knowledge of the system and the task domain as well as their individual cognitive strengths and limitations. Two important issues arise in building the user model: determining what information should be incorporated into the user model, and how this model should be configured.

Modeling the User

A user model is the description and knowledge of the user maintained by the system. In an adaptive interface, the user model varies from user to user and needs to be modified by the system as the individual user changes. The idea underlying an adaptive interface is that a specific user differs from other users and each individual user may change during the interaction with the system. The way in which users differ must be characterized, and accommodations to these differences must be built into the system. This information can then be used to build and maintain the user model.

User models vary in several different ways. Potosnak suggests that factor analysis can determine those factors that differentiate users with respect to the type of preferred interfaces [28]. These factors include computer experience, computer knowledge, and program specific knowledge. Thus, knowledge of the user can indicate the appropriate mode of interaction. When this knowledge is

incorporated into the model of a particular user, the system has a unique set of information to guide the interaction between the user and the system. Thus the interface becomes adaptive.

Because most of the research on adaptive computer interfaces is reported in the computer science literature, the work typically focuses on the software and implementation issues associated with building them. Consequently, most of the literature on adaptive interfaces usually addresses only the differences that arises from the experience levels of the users. That is, users are *novices* because of the limited experience with the system or computers. As users become more *computer proficient*, they become experts. Similarly, users may be classified as task domain novices if they do not have a rich base of knowledge of the domain or are classified as domain experts if they do.

Although these classifications are valid and important, they do not consider the inherent differences between people. Specifically, human cognitive or information processing skills and styles differ dramatically. Even among experts, different styles of interaction may call for different responses from the computer. This distinction may be more important for novices than for experts.

Clearly, cognitive psychology issues play a major role in modeling the user. If the system is to adapt to an individual user, it must encompass information abut users' cognitive limitations or strengths as well as users' perceptual strengths and weaknesses. In addition to these factors, users may differ in their style of interaction and in their preferences. Also, they may have a wide range of physical interaction preferences. This amount of information about the user should be modified as the user changes.

Cognitive psychology has an important bearing on user models because there are individual differences among users. For purposes of adaptive computer interfaces, the cognitive differences that arise among individuals must be characterized in logical classes. If the dimensions of the difference among individuals could be reliably identified, compensatory and accommodating modules could be incorporated into the interface. Two important dimensions on which computer users may differ are verbal and spatial abilities. Yallow conducted an investigation in which subjects with low and high spatial ability received material in one of two formats (graphic/spatial or verbal) [29]. The results suggest that immediate retention of material is better in a format in which the subject has high abilities. This seems to suggest that an interface may best facilitate users by presenting information in a form in which they have a stronger cognitive ability. Similarly, it may be better to present information in the same format with which the user has been trained.

Carbonell, in his development of an adaptive natural language interface, distinguishes between empirical and cognitive models that encompass information about the psychological structure of the user [30]. Although many adaptive systems do not attempt to include an empirical model of the user's knowledge, several do provide a cognitive model of the user. This is, some systems incorporate into the user model the knowledge the user possesses (the empirical model) as well as the user's internal reasoning strategies (the cognitive model). Both can serve as a basis for decisions concerning the user's goals or activities and direct how the system can assist. If a user's planning strategy is known, the adaptive interface can be more effective in assisting the user's problem solving strategies.

Robertson [31] and VanDerVeer et al. [32] note several cognitive styles that have a possible impact on human computer interaction. However, the various cognitive types that they delineate involve high-level processing and have not been explored in regard to adaptive interface systems.

Robertson also suggests that there may be individual differences in the way users distribute and allocate attentional resources. Differing attentional strategies must be more clearly defined so that the

system can adapt. Also, the system must compensate for user deficiencies; this may be especially important in systems that capitalize on windowing designs. Windows allow users to engage in multiple tasks or processes. Differential attentional capabilities among users may require different windowing strategies so that they can access all the necessary information.

Individuals also differ in their ability to direct their attention to various aspects of their task. Only a portion of users develop what is termed *cognitive tunnel vision* and are unable to attack a problem from a different angle. These users, however, must also be accommodated. For instance, a user may type the same incorrect command repeatedly because of the inability to realize that this is an inappropriate action. In the same situation, another user may have no problem shifting attention and producing the appropriate input.

Users may also differ in their planning strategies when trying to complete complex tasks. Goldin and Hayes-Roth suggest that there are distinct differences in actions taken by good planners and poor planners [33]. An adaptive system could accommodate and provide additional help to users who produce actions indicating poor planning strategies. This may be accomplished by restructuring the output that the user receives or by providing aids that help accomplish goals.

In addition to these higher cognitive functions, users may exhibit perceptual differences in their interaction with a system. This is an area that has not been included in the consideration of adaptive mechanisms in most systems.

One other issue that cannot be ignored is the user's mental model of the system. Any system, whether adaptive or not, should present the user with a coherent conceptual model of that system. Without this mental model the user may not be able to understand and integrate expectations of the system's behavior with respect to the actual behavior of the system [34]. Also, if there are a number of different mental models for a system, this must be taken into account in the design of the interface.

Constructing User Models

Several strategies are used to construct and modify user models. As noted earlier, the easiest technique for building user models is to classify users as novices and update their status to experts as they demonstrate more proficiency [11, 12, 14]. This is a simple task if there is a simple way of differentiating a novice from an expert. Mason and Thomas use some simple rules for classifying users into these categories [12]. They include the number of times the user has been on the system, the number of times the user has requested help, and the type of commands the user has invoked. These rules are assigned arbitrary weights. The user is upgraded to the next level when a predetermined threshold value is crossed. At each level, the interface behaves differently based on the assumptions of the user's system proficiency. This is, however, a very limited form of adaptation. It merely assumes that there are several levels or user types instead of just one.

In any real complex domain, a user probably progresses from novice to expert in a continuous fashion and not a stepwise manner. Accordingly, when the interface upgrades a user to a new level, this unexpected change in the system's behavior can be very jarring to the user. Maskery notes that error rates, performance times, and help requests increase dramatically as the system upgrades users through expertise levels [18]. Many users do not expect the change and do not understand why it has occurred. User maturity is slow and gradual. The progression from novice to expert cannot be characterized by a stepwise function. Although this is better than no adaptation, it certainly does not capture the differences that exist among all users. Any computer system should appear consistent and coherent to the users so they can focus their efforts on the primary task and not on the interface. If the system is not consistent, it should at least change in ways that the user can understand and interpret.

Another similar yet more differentiating technique of user modeling is to compare the user's knowledge to a domain expert's knowledge. That is, the current user knows some subset of the knowledge of the entire domain. This is most commonly found in tutoring systems where it is essential for the system to have a precise idea of the student's knowledge [35, 36]. It is assumed that the user *knows* something if the information or concept is used correctly. Further, it is assumed that the user knows additional concepts that must underly those that are used. If the concepts are used incorrectly or inappropriately, they are not part of the user's correct knowledge set. The most difficult and problematic part of this type of user model is that concepts that are not mentioned by the user may simply be unexpressed but not necessarily unknown.

This type of modeling is more powerful than a simple classification as *novice* or *expert* because it encompasses information about the knowledge set. Simple novice/expert classification models do not change if the user does something incorrectly; the model changes only if the user performs certain critical tasks correctly.

Examples of this type of user model are common in tutoring systems. Tutoring programs make extensive use of errors that student users make. Norman [37] and Matz [36] discuss the types of errors that are made by users attempting to satisfy plans and solve problems. Matz suggests that typical errors are not random and inconsistent but fall into three categories; the errors that are generated by an incorrect choice of extrapolation from prior or other information, the errors resulting from an incomplete but correct knowledge base, and the errors resulting from incorrect execution. This includes inputting commands in the wrong order, misplacing commands, and typographical errors. By classifying errors that are made by the user and the reasons for making them, underlying conceptual deficits can be uncovered and incorporated into a model for that user. This is exactly what the human tutor does during interaction with the student.

The stereotype model is a completely different kind of user model. In this approach, the user is characterized by a set of stereotypical traits. In GRUNDY, a program designed to recommend books, Rich asks the user to input a few self-descriptive words [15]. On the basis of this input, a set of traits is compiled that represents the user. This information is used to select books that should be appropriate for that user. This model is adaptive because it modifies its model of the user if its book choices are rejected.

Morik and Rollinger use a similar strategy in a system that is designed to recommend real estate [38]. In their system, users provide information that is used to make appropriate apartment selections with accompanying confidence ratings. Although this system performs well, its adaptive mechanism is designed only for matching in this context.

A robust modeling strategy must take into account more than can be inferred from a few initial user inputs. It must be able to change as the interaction continues and infer other information from the user's behavior. Information that a person volunteers is sometimes distorted and inaccurate. Also, a user may not be able to provide the pertinent information.

The stereotype model, however, has limitations. Indeed, the tasks chosen by Rich and by Morik and Rollinger are used because they lend themselves to this style of adaptive user representation. This user representation is useful when the system is doing matching but does not lend itself to other types of tasks such as tutoring and searching tasks. Although it captures some aspects of the user, it disregards others such as the user's proficiency with the system, or any knowledge the user has. Some of the appeal of this type of model is its similarity to the way that humans seem to characterize each other. Passini and Norman note that people assume a highly correlated default stereotype of others with whom they have virtually no prior knowledge and very little interaction [39]. Although these default sterotypes that people use may include a list of traits, they are much richer in information than GRUNDY's stereotypes.

Some systems integrate dialogue monitoring and stereotype generation. Finin and Drager have incorporated the stereotype model into their General User Modeling System, which they refer to as GUMS [16]. In addition to the initial inputs, their system uses other types of default logic in building the model. First, the sterotypes are arranged in a hierarchical tree in which lower levels provide more specific detail. In addition to the stereotypes, there are explicit default rules that cause facts about the user to be either asserted or assumed. Finally, GUMS uses failure as negation. This means that anything that is not able to be proven true is assumed false. This may work for a database that contains complete knowledge; however, in a more ambiguous or *open world*, this type of reasoning may not produce accurate information.

One final point needs to be made about constructing user models. Current user models do not address other issues. They include user idiosyncrasies, workload differences, cognitive capabilities, and individual preferences. Although some of these models encompass information relating to the user's proficiency, knowledge, and personality traits, none have provisions for user idiosyncrasies and preferences.

KNOWLEDGE OF THE INTERACTION

If an adaptive interface is to provide help that is appropriate to the context as well as to the particular user, it must be able to track the current human-computer dialogue. This requires some knowledge of how interactions are structured and what information may be implicit in them. This is most critical in natural language dialogues where the amount of implicit information can be very large. However, command languages do not free the adaptive system from needing to understand the current input as one piece of the interaction as a whole. Each action taken by the user must be interpreted in the context of the ongoing dialogue.

Natural Language Interfaces

Several adaptive interfaces have been implemented by using a natural language format [40, 19, 21, 41]. Natural language refers to the user's native language. Natural language interfaces are inherently more adaptive in that they do not require learning any artificial command syntax for communicating with the system. In a sense, any system that does not use natural language requires the user to adapt by learning artificialities of correct command formats or modes of interaction. Unfortunately, no natural language system can handle the wide range of inputs that are possible in natural language. This means that any natural language system still requires the user to adapt by restricting the legal inputs that are allowed.

Natural language interfaces possess many problems and difficulties that do not exist in other types of interfaces. Lehman and Carbonell cite the following criteria for natural language systems that are usable and friendly to novices and experts alike [42]:

- syntactic coverage—that is, it should be able to parse dialogue syntactically;
- task-oriented semantic coverage—that is, the interface should encompass a rich semantic knowledge of the domain to compensate for its restrictions on legal inputs;
- flexibility in the presence of extragrammaticality*—that is, the interface should be able to handle problems such as misspellings, transposed words, and missing punctuation [43];
- semantic resilience—that is, knowledge of the domain should be used to resolve ambiguities;

*This is a term coined by Carbonell and Hayes.

- user friendliness—that is, the interface should provide maximal assistance to the novice user and be unobtrusive to the expert;

- transportability—that is, the semantic domain knowledge should be separate from the interface itself so that the interface can be used in different domains.

These criteria are much easier to satisfy in a command language grammar because they are so restrictive. However, natural language brings with it an unrestricted domain of possible inputs. This is the biggest obstacle to implementing a natural language interface. If all possible inputs are not known beforehand, how are novel inputs to be processed? Where is the line between ungrammatical and nonsensical?

A natural-language interface must be more robust than a command-language interface. The information that can be derived from natural-language inputs is much greater than information from command languages. Linguists have long studied the nature of human dialogue and the implicit information in it. Many natural-language statements imply other information that is specified. Wahlster notes that assumptions about the user can be drawn from linguistic particles [21]. These information derivations cannot be done in a system that does not use natural language. The aspect of natural language that is most difficult to implement on a computer system is precisely that which allows it to process this implicit information.

However, some conversational norms enable conversants to extract implicit information. Reichman-Adar uses Gricean conversational principles* in an abstract computational natural language interface [19]. This interface works on several levels. Individual dialogue is analyzed. Also, the session as a whole is tracked and dialogue is interpreted in light of prior commands. Thus individual natural-language commands make up a coherent discourse. This is probably true for other non-natural-language interaction as well. However, natural-language discourses not only build upon previous input, they foster expectations concerning the next appropriate utterances. These expectations may serve to resolve ambiguities. Also, violated expectations are a rich source of information that is probably not available to static interfaces.

Most natural-language conversations center on the goals of the participants. These goals are not always explicitly specified and must be inferred from the implicit information that is contained in the conversation and situation. In a normal conversation between two humans, the hearer's task is to derive an explanation for the speaker's utterance. This derived explanation is usually based upon knowledge of the partner's beliefs and intentions. This information is used to produce an appropriate response. Ideally, a computer should use the same technique when interacting with users.

Because command languages are artificial, they can be designed to restrict the amount of implicit information that is carried in each user input. But this makes the system much less powerful. The restrictions of the command language constrain legal inputs and prevent novel inputs. Natural language interfaces, on the other hand, permit an infinite number of commands. This allows for different or novel approaches to the same problems or plans.

*Grician conversational principles are implicit *maxims* that constrain appropriate conversational moves. They include ideas such as: 1) make your contribution to the conversation as informative as required but not more informative than required; 2) make your contribution relevant; and 3) avoid obscurity, ambiguity, and excessive length. By assuming that dialogue participants adhere to these norms, assumptions can be made about their unstated goals and plans.

KNOWLEDGE OF THE TASK/DOMAIN

In most human-computer interaction, a user is trying to accomplish goals. These goals may be on several levels from the most immediate goals to the overall task goal. If a system is to be maximally supportive, it must be able to assist the user in achieving these goals. In most cases, whether the dialogue is conducted in an artificial language or a natural language, users do not explicitly state their goals. The system must be able to infer this information from the interaction. Only in this way will an adaptive system be able to provide the most appropriate assistance.

Task Modeling

Although many adaptive systems use a model of the user to gauge the amount and type of adaptation, several systems are not based upon user models. In this alternative approach, the adaptation is based upon the system's performance on the task. Greenberg and Witten report on their study in which an adaptive on-line telephone directory was developed [25]. Based on how frequently a telephone number is retrieved by an individual user, the structure of the telephone number database is reconfigured to make frequently recalled numbers easier to access. Thus, no information about the user is internalized in the system. The adaptation is based solely on the past performance with the task.

Croft, who used a document retrieval task as well, provides another example [22]. Each search is rated for its effectiveness. An associative search network (ASN) is used to reinforce *good* searches. Barto, Sutton, and Brouwer provide an excellent discussion of the mechanics of the ASN [45]. Again, the adaptation is based on system performance and not on any characteristics of the user.

Goal Detection and Plan Inference

An adaptive interface must know what the user wants to accomplish. This entails detecting the plans that the user holds for realizing the task goals. In human-computer interaction, there are two possible conditions under which plan recognition occurs. Each requires a different plan of recognition strategy. First, there is the case in which all possible plans of the user are known. This occurs when the task domain is limited and only a certain number of alternatives are available to a user. Second, there is the case in which all possible plans are not known. This is usually the case in any reasonably complex system.

If all possible plans are known, the system simply searches through them and selects the one that is the closest match to the actions that are performed by the user. If all possible plans are not specified, the system first can search through the plans for a close match. If it does not find a close match, it can use several strategies. Most easily and most unsatisfactorily, it can do nothing and give up. Otherwise, the system can ask the user about the plan and add the user's new plan to its internal plan list. This may not be an optimal solution either, because the user may not be able to verbalize the plan or may not be able to convey it to the system. Also, forcing the user to state a plan interrupts current activities.

If the system does not interrogate the user, it must deduce the user's plan from the situation and its current knowledge of the user. The goal of the system is to provide cooperative behavior for the user. To provide the appropriate behavior, the system must deduce the user's plan and what action the system must take for the plan to be satisfied. If the user is making a direct request for information or action, the system response is very obvious. However, many user plans may not be directly specified or implied. Therefore, the plan must be inferred from an *indirect speech act* [40].

Indirect speech acts are very pervasive in human communication. A typical example is the question "Do you know the time?" The appropriate response is not yes, although this is a question requiring a yes/no answer. A person typically responds with the time because of the underlying assumption that the person's plan or goal is to know the time. Allen and Perrault [40] as well as Wilensky, Arens, and Chin [41] discuss indirect speech acts and plan recognition in the context of natural language. This issue has an important bearing on natural language HCIs. The reason is that when a command language is used, many forms of direct speech acts may not be legal inputs. The assumptions made about the users goals drive the interpretation of the indirect speech act. It should not be taken at face value and treated literally. In the case of the question posed above, the user would not be helped by the answer, "Yes, I know the time. "

Wilensky et al. Unix Consultant (UC), which is a natural language help system for Unix, attempts to recognize the users' goals from their inputs [41]. It tries to construct a plan that the user may hold and that satisfies the goal. The UC system produces a response that helps the user solve the goal. Although UC takes the immediate context into account, it does not have a model of the current user. Therefore, it only provides an answer unique to the situation but not necessarily unique or optimal for the particular user.

The context of the interaction encompasses not only the previous dialogue but also the environment in which the user is working. This creates some problems for users in computer systems that use windows. Reichman-Adar has examined the parallel between context in verbal communication and in human computer interaction in windowing systems [46]. This study suggests that users perceive windows distinct from each other but consistent within each single window. Thus, the user bases actions upon these unstated assumptions. The user's implicit perceptions of the system need to be taken into account in the system design.

Reichman-Adar also suggests that the system should act as a *smart assistant* who may interrupt the user but should not interrupt at the wrong time. Therefore, the system must keep track of what the user is doing in order to interrupt at task boundaries or suitable interim points. This again raises the point that the system must be able to detect the user's goals.

It is important to note that with respect to adaptive HCIs, it can safely be assumed that every input made by the user is intended to convey some information to the system. Also, it is assumed that the user does not *chit chat* with the system but tries to accomplish a goal in the shortest amount of time. Any input can be viewed as either an attempt to gain information or an attempt to direct the system to help the user attain other goals.

Help Systems

The purpose of any help system is to assist the user. Consequently, these computer interfaces can be significantly enhanced by adapting to the individual user. Each particular user has different problems based on goals, knowledge of the task domain, and familiarity with the interaction environment. Consequently, users differ in the type and amount of help that is needed.

One way to customize the help facility to an individual user is to embed examples specific to the user in *canned* examples that are provided to all users [13]. Even this strategy requires system knowledge of the domain, the user, and the current context.

Mason and Thomas provide a prototype of an adaptable on-line help manual for Unix [12]. Users requesting information from the manual are provided with different amounts of information. This information contains more sophisticated and extensive material for users classified as more *expert* or system proficient. However, their system does not take the context of the help request into account.

The system must know what the user is trying to accomplish in order to provide the appropriate type of help. In the case of a user who is directly asking for help, the system must deduce what information the user is seeking. This seems straightforward if the user asks a question that simply needs a direct answer. However, when human experts provide advice they do not always respond directly to the question that is asked. A study of user queries to an expert on electronic mail reports that the human experts attempt to infer the advice-seekers plan and provides an answer that helps them achieve that goal [20]. Thus, the answer provided by the expert is tailored to the user and the situation. Typical help systems do not take the context of the query into account and therefore cannot modify their answers to suit the particular user and the current situation.

Fischer, Lemke, and Schwab have developed two related knowledge-based help systems [47]. ACTIVIST and PASSIVIST, respectively, are active intervening and passive request-driven help systems designed to provide assistance on the Unix system. PASSIVIST takes natural language questions or requests for help and interprets them in light of the current context of the user. That is, in light of what the user is doing or attempting to do, the system tries to deduce what information the user is seeking with the help request.

ACTIVIST is an active help system that monitors user behavior and intervenes when it detects the user performing below an optimal level. This performance can be of two kinds. First, the user invokes several commands to accomplish something that could be done with less commands. Second, the user does not know the minimal amount of keystrokes for a command. For example, the user types the full command name when only one function key is needed. Here again, the system must infer what the user is trying to do and provide the appropriate information. On the other hand, a help system that bombards the user with help messages ceases to be helpful and simply becomes annoying. Also, the user may have some reason, unknown to the system, for using the longer commands. In this case, help messages are definitely counterproductive and bothersome.

KNOWLEDGE OF THE SYSTEM

In addition to knowledge about the user, an adaptive system should have knowledge of itself. To provide the user with the most support, the system must be aware of its own strengths and limitations. Although input and output should be tailored to the current task and user's needs, the capabilities of the system impose limitations on what type of restructuring can be done. The system must be able to optimize the input and output within the boundaries of its inherent limits.

Input/Output Issues

The best computer support tool is one that is easy to use and powerful in capabilities. The ease of use of any adaptive system is tied very closely to the input/output capabilities of the system. As mentioned previously, natural language interfaces are inherently more adaptive than command language interfaces because they do not require the user to adapt to the system's language.

Input and output modes that are chosen for a particular system should reflect the user's limitations and capabilities as well as the particular task for which the system is designed. Card, English, and Burr suggest that the mouse is the most efficient device (in comparison to a joystick, step keys, and function keys) for selecting a field on the CRT screen [48]. Thus the mouse should be the input device of choice for this task. However, for other tasks, another input device may be superior.

Similarly, the type of output should be dictated by the limits and capabilities of the user, the task type, and the displayed information. The human operator has certain cognitive limits that must be

addressed by the system output. Data should be displayed in a way that facilitates easy scanning, perception, and interpretation [49]. This may be accomplished in any number of ways. The data itself may be restructured. The internal structure of the data may be configured on the screen. Color or highlighting can be used, or other more sophisticated pictorial graphics may be employed. The type of output is highly dependent on the nature and purpose of the information.

Also, the information should be specifically tailored to the strengths and weaknesses of the particular user. That is, the current user may differ in the strategies or approaches depending on the information. An ideal adaptive interface compensates for weaknesses of a user's information processing strategies and exploits the user's strengths.

CONCLUSIONS

As computer systems become increasingly more complex, the need for an interface that can adapt to the current user and context becomes crucial. Historically, it has been the user who has had to adapt to the system. As system and task complexity increases, user performance is degraded if users change their behavior to suit the system. This procedure should be done by the system so that users can focus on their primary tasks. The interface should free the user from system specific details and provide as much support as possible to help achieve goals.

Although several attempts have been made to construct and implement adaptive interfaces, most have not addressed crucial design issues. The adaptive interface must encompass knowledge of the interaction, system, task domain, and most importantly, the user. Only with these types of knowledge will the interface be able to augment performance on an individual basis. Improved performance has been exhibited in systems with these adaptive mechanisms, but the range of adaptive behavior in them has been quite narrow. Typically, the interface includes only some of the necessary knowledge. For example, natural-language interfaces such as Reichman-Adar's include knowledge of interaction and dialogue but not information concerning the user or task domain. Alternately, adaptive help systems like Fisher, Lemke, and Schwab's include knowledge specific to the domain but do not incorporate knowledge that is specific to the user. Those systems that do include user specific information do not include the other critical types of information.

Overall, the most neglected component of computer systems has been the human user. To produce the most effective human-computer system, efforts must be made to delineate important information concerning both the human and computer components. Although there has been considerable advancement in understanding hardware aspects of systems, there has been considerably less advancement in understanding the human aspects of systems. The user's cognitive strengths and limitations must be incorporated into the sytem's knowledge base. This direction warrants increased research efforts.

ACKNOWLEDGMENTS

The authors thank L. J. Chmura whose critical reading and review of this report substantially enhanced its presentation. Also, the authors are indebted to Lisa Achille and Keith Kramer, who offered insights, suggestions, and refinements to our approach.

REFERENCES

1. L. Kleinrock, "Distributed System," *Comm. ACM* **28**(11), 1200-1213 (1985).

2. C.D. Rose, "R and D Race Tightens for 5th-Generation Computers," *Electronics* 30-31 (1985).

3. Human Engineering Criteria for Military Systems, Equipment and Facilities, MIL-STD 1472C, Department of Defense, Washington, DC (1981).

4. S.L. Smith and J.N. Mosier, "Design Guidelines for the User-System Interface Software," The Mitre Corporation Technical Report ESD-TR-84-190, Bedford, MA, 1984.

5. D.V. Morland, "The Evolution of Software Architecture," *Datamation,* 123-132 (1985).

6. J. Meads, "Report on the SIGCHI Workshop on Planning for User Interface Standards," *SIGCHI Bull.* **17**(2), 11-16 (1985).

7. S.K. Card, T.P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction,* Lawrence Erlbaum Associates, Hillsdale, NJ, (1983).

8. D.L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules," *Comm. ACM* **15**, 1053-1058 (1972).

9. B. Liskov and S. Zilles, "Specification Techniques for Data Abstractions," *IEEE Tran. on Software Eng.* **SE-1**(1), 7-19 (1975).

10. W. Buxton, M.R. Lamb, D. Sherman, and K.C. Smith, "Towards a Comprehensive User Interface Management System," *Computer Graphics* **17**(3), 35-42 (1983).

11. E.A. Edmonds, "Adaptive Man-Computer Interfaces," pp. 389-426 in *Computing Skills and the User Interface*, M.J. Coombs and J.L. Alty, eds. (Academic Press, London, 1981).

12. M.V. Mason and R.C. Thomas, "Experimental Adaptive Interface," *Information Technology: Research, Design, Applications* **3**(3), 162-167 (1984).

13. E. Rissland, "Ingredients of Intelligent User Interfaces," *Int. J. Man-Machine Studies* **21**, 377-388 (1984).

14. H. Mozeico, "A Human/Computer Interface to Accommodate User Learning Stages," *Comm. ACM* **25**(2), 100-104 (1982).

15. E. Rich, "User Modeling via Stereotypes," *Cognitive Science* **3**, 329-354 (1979).

16. T. Finin and D. Drager, "GUMS: A General User Modeling System," University of Pennsylvania School of Engineering and Applied Science, Technical Report MS-CIS-86-35, May 1986.

17. I. Monarch and J. Carbonell, "CoalSORT: A Knowledge-Based Interface,"*IEEE Expert* **2**(1), 39-53 (1987).

18. H.S. Maskery, "Adaptive Interfaces for Naive Users—An Experimental Study," *Interact '84, Proc IFIP Conf.*, 343-350 (1984).

19. R. Reichman-Adar, "Extended Person-Machine Interface," *Artificial Intelligence* **22**, 157-218 (1984).

20. M.E. Pollack, "Information Sought and Information Provided: An Empirical Study of User/Expert Dialogues," *Proc. CHI 85 Human Factors in Computing Systems,* 155-159 (1985).

21. W. Wahlster and A. Kobsa, "Dialogue-Based User Models," *Proc. IEEE* **74**(7) 948-960 (1986).

22. W.B. Croft, "The Role of Context and Adaptation in User Interfaces," *International J. of Man-Machine Studies* **21**, 283-292 (1984).

23. W., Sherman, "SAUCI: Self-Adaptive User-Computer Interfaces," Ph.D. dissertation, University of Pittsburgh, Pittsburgh, PA, 1986.

24. D.L. Parnas, "Use of Abstract Interfaces in the Development of Software for Embedded Systems," NRL Report 8047, 1977.

25. S. Greenberg and L.H Witten, "Adaptive personalized Interfaces—A Question of Viability," *Behavior and Information Technology* **4**(1), 31-45 (1985).

26. W.B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems," *Computing Surveys* **13**, 71-100 (1981).

27. A. Freedy, K.B. Davis, R. Steeb, M.G. Samet, and P.C. Gardiner, "Adaptive Computer Aiding in Dynamic Decision Processes: Methodology, Evaluation, and Application," ARPA Technical Report PFTR-1016-76-8/30, 1976.

28. K.M. Potosnak, "Choice of Interface Modes by Empirical Grouping of Computer Users," *Proc. IFIPS*, 27-32 (1984).

29. E. Yallow, "Individual Differences in Learning from Verbal and Figural Materials," School of Education, Standford University, Aptitudes Research Project Technical Report No. 13, Palo Alto, CA, 1980.

30. J.G. Carbonell, "The Role of User Modeling in Natural Language Interface Design," Computer Science Technical Report CMU-CS-83-115, Carnegie-Mellon University, Pittsburgh, PA, 1983.

31. I.T. Robertson, "Human Information-Processing Strategies and Style," *Behavior and Information Technology* **4**(1), 19-29 (1985).

32. G.C. VanDerVeer, M.J. Tauber, Y. Waern, and B. VanMuylwijk "On the Interaction Between System and User Characteristics," *Behavior and Information Technology* **4**(4), 289-308 (1985).

33. E.E. Goldin and B. Hayes-Roth, "Individual Differences in Planning Processes," Office of Naval Research Note # N-1488-ONR, 1980.

34. D.A. Norman, "Some Observations on Mental Models," pp. 7-14 in *Mental Models,* D. Gentner, and A.L. Stevens, eds. (Lawrence Erlbaum Associates, Inc., Hillsdale, N.J. 1983).

35. R.R. Burton and J.S. Brown, "A Tutoring and Student Modeling Paradigm for Gaming Environments," *Proc. ACM SIGCSE/SIGCUE Joint Symposium* (1976).

36. M. Matz, "Towards a Process Model for High School Algebra Errors," in *Intelligent Tutoring Systems,* D. Sleeman and J.S. Brown, eds. (Academic Press, New York 1982).

37. D.A. Norman, "Design Rules Based Upon Analyses of Human Error," *Comm. ACM* **26**, 254-258 (1983).

38. K. Morik and C., Rollinger, "The Real Estate Agent-Modeling Users by Uncertain Reasoning," *The AI Magazine* 44-52 (1985).

39. F.T. Passini and W.T. Norman, "A Universal Conception of Personality Structure," *J. Personality and Social Psychology* **4**, 44-49 (1966).

40. J.J. Allen and C.R. Perrault, "Analyzing Intention in Utterances," *Artificial Intelligence* **15**, 143-178 (1980).

41. R. Wilensky, Y. Arens, and D. Chin, "Talking to Unix in English: An overview of UC," *Comm.* ACM **27**, 574-593 (1984).

42. J.F. Lehman and J.G. Carbonell, "Learning the User's Language: A Step Towards Automated Creation of User Models," (Carnegie-Mellon University, Pittsburg, PA 1987).

43. J.G. Carbonell and P.J. Hayes, "Recovery Strategies for Parsing Extragrammatical Language," Computer Science Technical Report CMU-CS-84-107, Carnegie-Mellon University, Pittsburgh, PA 1984.

44. H.P. Grice, "Logic and Conversation," pp. 41-58 in *Syntax and Semantics,* P. Cole and J. Morgan, eds. (Academic Press, New York, 1975).

45. A.G. Barto, R.S. Sutton, and P.S. Brouwer, "Associative Search Network; A Reinforcement Learning Associative Memory," *Biological Cybernetics* **40**, 201-211 (1981).

46. R. Reichman-Adar, "Communication Paradigmas for a Window System," pp. 285-313 in *User Centered System Design: New Perspectives in Human-Computer Interactions*, D.A. Norman and D.W. Draper, eds., (Lawrence Erlbaum Associates, Hillsdale, NJ, (1986).

47. G. Fischer, A. Lemke, and T. Schwab, "Knowledge-Based Help Systems," *Proc. of CHI 85 Human Factors in Computing Systems,* 161-167 (1985).

48. S.K. Card, W.K. English, and B.J. Burr, "Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys and Text Keys for Text Selection on a CRT," *Ergonomics* **21**(8), 601-613 (1978)

49. A. Morse, "Some Principles for the Effective Display of Data," *Comm. ACM* 94-101 (1979).