INFORMATION TECHNOLOGY FOR COMMAND AND CONTROL

Methods and Tools for Systems Development and Evaluation

Edited by

Stephen J. Andriole
Drexel University

Stanley M. Halpin
Army Research Institute

A volume in the IEEE PRESS Selected Reprint Series,
prepared under the sponsorship of the
IEEE Systems, Man, and Cybernetics Society.

The Institute of Electrical and Electronics Engineers, Inc., New York

IEEE PRESS
445 Hoes Lane, PO Box 1331
Piscataway, NJ 08855-1331

ISBN 0-87942-270-X

IEEE Order Number: PC0264-2

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Information technology for command and control : methods and tools for systems development and evaluation / edited by Stephen J. Andriole, Stanley M. Halpin.
p. cm. -- (IEEE Press selected reprint series)
"Prepared under the sponsorship of the IEEE Systems, Man, and Cybernetics Society."
Includes bibliographical references and index.
ISBN 0-87942-270-X (casebound) :
1. Command and control systems. 2. Management—Communication systems. 3. Management—Data processing. 4. Decision support systems. I. Andriole, Stephen J. II. Halpin, Stanley M.
III. IEEE Systems, Man, and Cybernetics Society.
UB212.I55 1991
355.3'3041'028--dc20 90-20526
CIP

Adaptive Human–Computer Interfaces: A Literature Survey and Perspective

ANTHONY F. NORCIO, MEMBER, AND JAKI STANLEY

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

I. INTRODUCTION

MANY computer applications, people are frequently sented with exhaustive amounts of data and must critical decisions in brief time intervals. Such situa- are often made worse because users lack the knowl- and experience to be effective. Because of this and se hardware and applications are becoming more ex, the area of human–computer interface (HCI) ology is becoming essential to the operational success ny systems. Unfortunately, the technology to support evelopment and implementation of effective user ıter interfaces is lacking. At least one authority warns, ss we pay close attention to the user interface, users ecome hopelessly lost and ineffective... the complex- the systems should be transparent to users" [18]. To score the problem further, DARPA[1] officials now ıd that interface research is lagging while other 5th ıtion research efforts are progressing [42]. Some HCI ines exist [1], [45], but they are not designed to le the necessary insights for determining the effec- ss and appropriateness of specific interfaces.

re are at least three major factors underlying the quacy of HCI technology. The first is that interface ıre is generally not viewed as part of the system but as a software package between the system and the 27]. This traditional approach maintains both the ıce and the user external to the system rather than f it. This results in 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 human–computer interface technology is that the design of effective interfaces is a difficult problem with sparse theoretical foundations [24]. Though components of a theoretical framework have been suggested [9], 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 moderately enhanced at best. The old adage that there is nothing more practical than a good theory is appropriate here. Care must be taken not to view technology enhancement as technology advancement. It is difficult to imagine significant HCI technology advancements until a sound theoretical foundation is established.

And finally, the third 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 [32] in an abstract interface [20] 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) [5]. The thrust of the UIMS approach is to make human–computer interfaces 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. This work needs to be extended by incorporating cognitive models of users that are abstract specifications of user populations.

An extension of UIMS is the concept of an adaptive interface [11], [22]. The idea of an adaptive interface is straightforward. Simply, it means that the interface should adapt to the user; rather than the user adapting to the system. In spite of this apparent simplicity, the problems that are implicit in adaptive interfaces are fundamentally difficult and complex [40]. One of the first issues that needs to be addressed is the problem of user models [29], [39]. Recent studies have shown that user models that underlie adaptive systems must necessarily be based on theories of cognition and must explain evolving changes in user performance and capability [12], [25]. The purpose of the models is to deduce users levels of expertise and experience by collecting input parameters such as command types, error rates, and speed [21], [37].

ıscript received June 30, 1988; revised November 20, 1988.
. Norcio is with the Information Systems Management Depart- Jniversity of Maryland, Catonsville, MD 21228.
anley is with the Psychology Department, Catholic University, gton, DC 20069.
. Log Number 8826386.
ense Advanced Research Projects Agency.

Reprinted from *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 2, pp. 399–408, Mar./Apr. 1989.

Another critical aspect of an adaptive interface is the dialogue between the user and the system [40]. The dialogue must be appropriate for the specific user [35], [47]. In addition, it is suggested that the application plays a major role in the success of adaptive interfaces [10]. It appears, however, that no study has attempted to examine these inter-related issues in the context of a unified approach.

Equally important is the structure and architecture of the interface [44]. 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 (e.g., [33]) need to be explored for designing adaptive interfaces.

A. Background

The major focus of system design traditionally concerns the functionality of the system. 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. Currently, the importance of the interface is gaining more attention. This is evident in the novel and extremely effective interface designs that have surfaced. These include the use of different input/output (I/O) devices such as the mouse, the light pen, the touch screen as well as innovative presentation methods such as windows and icons.

These developments make computer systems easier to use by a larger variety of users. 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 can 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 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, providing help appropriate to the context, and decreasing the mental and physical workload of that particular user.

An interface that can be adapted to the user would necessarily be more complex than one that cannot. There are two ways that a system can be adaptive. The first way is to leave the interface in a form that enables modification by the user if the behavior of the system is judged unsatisfactory once it is in operation. Edmonds [11] discusses interfaces that may be modified by several different classes of people. The interface may be modified by a computer specialist, a trained user, or any user. The amount of change that is allowed depends on the user who is making the modifications and the access privileges that they 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 paper an interface that dynamically modifies itself is what is meant by the term *adaptive interface*. An adaptive interface needs to have informati that is generally not required or available to a sta interface. Recently research interest has increased in area of system adaptation through machine learning. A form of machine learning can be termed adaptation cause the machine assimilates new information and sponds more appropriately to new and novel situatio However, machine learning does not always constitute adaptive interface. The adaptive interface changes w respect to the particular user and current context whil machine that can "learn" may behave the same way w all users. Thus, an adaptive system only becomes an ada tive interface when it learns with respect to the individu user and not when it learns only with respect to the ta domain. That is, an adaptive interface works differen depending on the current context. This includes both current task and the current user.

B. Negative Aspects of Adaptive Human–Computer Interfaces

The concept of an adaptive interface does have criti Greenberg and Witten suggest some reasons why they m not be desirable [16]. First, the user may not be able develop a coherent model of the system if the system frequently changing. This may undermine the user's con dence and performance with the system. If the user do not have a clear understanding of the system behavi user effectiveness can be seriously reduced.

Another problem that may arise with an adaptive syst is the loss of control or the feeling of loss of control th the user may experience. Wahlster and Kobsa note th users may attempt to disguise their goals and preferen [47]. They suggest that the interface should allow users inspect a comprehensible version of their models. In ad tion, the system should allow users the option of "turn off" the modeling component of the interface. Howev an adaptive interface is not designed to take control fr the user. But rather, it is intended to provide the maxim and most appropriate assistance to a given user for current task.

Another disadvantage to adaptive interfaces is an crease in implementation complexities and costs. Althou this may be a problem now, as development of adapt interfaces continues, the cost of implementation sho decrease. In addition, adaptive interfaces also incu higher computational overhead. Any interface that ha complex modeling component must do more computat than a system that does no modeling. This uses compu tional resources and consequently may increase the sys response time. But the advances in hardware technol are resulting in lower costs and faster systems.

C. Positive Aspects of Adaptive Human–Computer Interfaces

On the other hand, there are several positive aspect adaptive interfaces. An area that lends itself well to need for an adaptive interface is system automation system that dynamically allocates tasks must be abl

adapt to individual users. Rouse suggests that tasks should be allocated to both the user and the computer [43]. 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 necessarily 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 people vary in the amount and type of information they can process as well as the way in which they process it.

For example, when human operators are confronted with an overwhelming amount of information in a decision making task, they must decide what information to request and use. In such environments, it has been found that human operators usually do not produce optimal behavior compared to a Bayesian normative model of the task [14]. 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.

There are other reasons to have the computer do the adapting. Many times users may not have the necessary information or expertise to modify their behavior. Indeed, they may not know that modifications can or need be 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. In addition, an adaptive system increases user proficiency with a new system and prevents frustration with an overly simple system [16].

The issue is whether the advantages of adaptive interfaces outweigh the costs associated with constructing and executing them. However, the main criticisms 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.

D. Knowledge Required by an Adaptive Interface

An adaptive interface needs to include a knowledge-base that encompasses four domains. These are [10], [40]:

1) knowledge of the user; that is, expertise with the system;
2) knowledge of the interaction; that is, modalites of interaction and dialogue management;
3) knowledge of the task/domain; that is, the ultimate purpose of the problem area and its goals; and
4) knowledge of the system; that is, the system's characteristics.

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

II. Knowledge of the User

An adaptive system must be able to characterize and distinguish between individuals. A user model, combining 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. There are two important issues that arise in building the user model: determining what information should be incorporated into the user model and how this model should be configured.

A. 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 modifiable 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 [36]. These factors include computer experience, computer knowledge, and program specific knowledge. Thus, knowing 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. In this way, 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 only addresses the differences that arise from experience levels of the users. That is, users are "novices" due to 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 take into consideration the differences between people that may be inherent. Specifically, human cognitive or information processing skills and styles differ dramatically. Even among experts there are differing interaction styles that 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 about users' cognitive limitations or strengths as well as users' perceptual strengths and weaknesses. In addition to these factors, users may differ in their interaction style and preferences. Also, they may have a wide range of physical interaction preferences. This store 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 between users. For purposes of adaptive computer interfaces, the cognitive differences that arise between individuals must be characterized in logical classes. If dimensions on which individuals differ can be reliably identified, compensatory and accommodating modules can 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) [49]. The results suggests 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, make a distinction between empirical and cognitive models that encompass information about the psychological structure of the user [6]. 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. That 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 [41] and VanDerVeer *et al.* [46] 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. In addition, 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 windo strategies so that all necessary information can be cessed.

Individuals also differ in their ability to direct t attention to various aspects of their task. Only a portio users develop what is termed "cognitive tunnel vision" are unable to attack a problem from a different angle. these users must also be accommodated. For instanc user may type the same incorrect command repeatedly to an inability to realize that this is an inappropr action. In the same situation, another user may have problem shifting attention and producing the appropr input.

Users may also differ in planning strategies when try to complete complex tasks. Goldin and Hayes-Roth s gest that there are distinct differences in actions taken good planners and poor planners [15]. An adaptive sys could accommodate and provide additional help to u who produce actions indicating poor planning strateg This may be accomplished by restructuring the output t the user receives or by providing aids that help accomp goals.

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

One other issue that can not be ignored is the us mental model of the system. Any system, whether adapt or not, should present the user with a coherent concept model of that system. Without this mental model the u may not be able to understand and integrate expectati of the system's behavior with respect to the actual behav of the system [30]. Also, if there are a number of differ mental models for a system, this must be taken i account in the design of the interface.

B. Constructing User Models

Several strategies are used to construct and modify u models. As noted earlier, the easiest technique for build user models is to classify users as novices and update th status to experts as they demonstrate more proficie [11], [22], [29]. This is a simple task if there is a simple v of differentiating a novice from an expert. Mason a Thomas use some simple rules for classifying users i these categories [22]. They include the number of times user has been on the system, the number of times the u has requested help, and the type of commands the user i invoked. These rules are assigned arbitrary weights. T user is upgraded to the next level when a predetermi threshold value is crossed. At each level, the interf behaves differently based on assumptions of the us system proficiency. This is, however, a very limited form adaptation. It merely assumes that there are several lev or user types instead of just one.

In any real complex domain, a user probably progres from novice to expert in a continuous fashion and no

step wise one. 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 [21]. 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 can not be characterized by a stepwise function. Although this is better than no adaptation, it certainly does not capture the differences that exist between all users. Any computer system should appear consistent and coherent to 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 in which it is essential for the system to have a precise idea of the student's knowledge [4], [23]. A user is assumed to "know" something if the information or concept is used correctly. Further, the user is assumed to know 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 [31] and Matz [23] 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. First, are those errors that are generated by an incorrect choice of extrapolation from prior or other information. Next are the errors resulting from an incomplete but correct knowledge base. The third type of errors result 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 them, underlying conceptual deficits can be uncovered and incorporated into the model for that user. This is exactly what the human tutor does during interaction with the student.

A completely different kind of user model is the stereotype 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 [39]. On the basis of this input, a set of traits that represent the user is compiled. 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 [26]. 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 [34]. Although these default stereotypes that people use may include a list of traits, they are much richer in information than Grundy's stereotypes.

Some systems do 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 [12]. In addition to the initial inputs, their system uses other types of default logic in building the model. First, the stereotypes 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 about constructing user models needs to be made. There are other issues that current user models do not address. 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.

III. Knowledge of 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.

A. Natural Language Interfaces

Several adaptive interfaces have been implemented using a natural language format [2], [37], [47], [48]. 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, there is no natural language system that 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 with 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 [19].

1) Syntactic coverage; that is, it should be able to parse dialogue syntactically.
2) 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.
3) Flexibility in the presence of extragrammaticality[2], that is, the interface should be able to handle problems such as misspellings, transposed words, missing punctuation etc. [7].
4) Semantic resilience; that is, knowledge of the domain should be used to resolve ambiguities.
5) User friendliness; that is the interface should provide maximal assistance to the novice user and be unobtrusive to the expert.
6) Transportability; that is, the semantic domain knowledge should be separate from the interface itself so that the interface can be used in different domains.

[2]This is a term coined by Carbonell and Hayes.

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 than is specified. Wahlster notes that assumptions about the user can be drawn from linguistic particles [47]. 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, there are some conversational norms that enable conversants to extract implicit information. Reichman-Adar uses Gricean conversational principles[3] in an abstract computational natural language interface [17], [37]. This interface works on several levels. Individual dialogue is analyzed. In addition, 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 nonnatural 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. This is one 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

[3]Gricean 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.

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.

IV. 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 this way will an adaptive system be able to provide the most appropriate assistance.

A. Task Modeling

Although many adaptive systems use a model of the user to gauge the amount and type of adaptation, there are several systems that 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 [16]. 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 [10]. 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 [3]. Again, the adaptation is based on system performance and not on any characteristic of the user.

B. 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 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 there are only a certain number of alternatives that a user can possibly be attempting to accomplish. 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 all the possible plans and selects the one that is the closest match to the actions that are performed by the user. If it is known that all possible plans are not specified, the system can first search through the plans for a close match. If it does not find a close match, there are several strategies that it can use. 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. In order to provide the appropriate behavior, the system must deduce the user's plan and what action the system must take in order 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. In this case, the plan must be inferred from an "indirect speech act" [2].

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 respnds with the time because of the underlying assumption that the person's plan or goal is to know the time. Allen and Perrault [2] as well as Wilensky, Arens and Chin [48] discuss indirect speech acts and plan recognition in the context of natural language. This issue has an important bearing on natural language human–computer interfaces. The reason for this is that, when a command language is used, many forms of indirect speech acts may not be legal inputs. The assumptions made about the user's 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.'s* unix consultant, which is a natural language help system for Unix, attempts to recognize the user's goals from their inputs [48]. It tries to construct a plan, which the user may hold, 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 is limited to providing 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 utilize windows. Reichman-Adar has examined the parallel between context in verbal communication and in human–computer interaction in windowing systems [38]. This study suggests that users per-

ceive 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 human–computer interfaces, it can safely be assumed that every input made by the user is intended to convey some information to the system. This assumes also, that the user does not "chit chat" with the system but is trying 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.

C. 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 differing problems based on their goals, their knowledge of the task domain, and their 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 [40]. 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 [22]. Users requesting information from the manual are provided 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 the human experts attempt to infer the advice-seeker's plan and provide an answer that help them achieve that goal [35]. 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 cu situation.

Fischer, Lemke, and Schwab have developed two rel knowledge-based help systems [13]. Activist and Passi respectively, are active intervening and passive req driven help systems designed to provide assistance on Unix system. Passivist takes natural language question requests for help and interprets them in light of the cur context of the user. That is, in light of what the use doing or attempting to do, the system tries to deduce w information the user is seeking with the help request.

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

V. Knowledge of the System

In addition to knowledge about the user an adapt system should have knowledge about itself. In order provide the user with the most support the system must aware of its own strengths and limitations. Although inp and output should be tailored to the current task a user's needs, the capabilities of the system impose limi tions on what type of restructuring can be done. T system must be able to optimize the input and outp within the boundaries of its inherent limits.

A. Input/Output Issues

The best computer support tool is most likely one tha easy to use and powerful in its capabilities. The ease of of any adaptive system is tied very closely to input/output capabilities of the system. As mentior previously, natural language interfaces are inherently m adaptive than command language interfaces because th do not require the user to adapt to the system's langua

Input and output modes that are chosen for a particu system should reflect the user's limitations and capabilit as well as the particular task for which the system designed. Card, English, and Burr suggest that the mou is the most efficient device (in comparison to a joysti step keys and function keys) for selecting a field on CRT screen [8]. Thus, the mouse should be the inp device of choice for this task. However, for other tas 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 [28]. This may be accomplished in any number of ways. The data itself may be restructured. The data may also be configured on the screen to reflect its internal structure. 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.

In addition, 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 any weaknesses that a user's particular information processing strategies contain and exploits the user's strengths.

VI. Conclusion

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 increase, user performance will be degraded if users must change their behavior to suit the system. This 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 in achieving the 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 knowledge of interaction and dialogue, but not information concerning the user or task domain. Alternately, adaptive help systems like Fischer, 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 user, that is, the human user. In order to produce the most effective human–computer system, there must be efforts to delineate important information concerning both the human and computer components. While there has been considerable advancement in understanding hardware aspects of systems, there has been considerably less advancement in understanding of the human aspects of systems. The user's cognitive strengths and limitations must be incorporated into the system's knowledge base. This is a direction that warrants increased research efforts.

Acknowledgment

The authors would like to thank L. J. Chmura whose critical reading and review of this paper substantially enhanced its presentation. In addition, the authors are indebted to Lisa Achille and Keith Kramer, both of whom offered insights, suggestions, and refinements to our approach.

References

[1] *Human Engineering Criteria for Military Systems, Equipment and Facilities*, MIL-STD 1472C, Department of Defense, Washington, DC, 1981.

[2] J. J. Allen and C. R. Perrault, "Analyzing intention in utterances," *Artificial Intell.*, vol. 15, pp. 143–178, 1980.

[3] A. G. Barto, R. S. Sutton, and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biological Cybern.*, vol. 40, pp. 201–211, 1981.

[4] R. R. Burton and J. S. Brown, "A tutoring and student modeling paradigm for gaming environments," in *Proc. ACM SIGCSE/SIGCUE Joint Symp.*, 1976.

[5] W. Buxton, M. R. Lamb, D. Sherman, and K. C. Smith, "Towards a comprehensive user interface management system," *Computer Graphics*, vol. 17, no. 3, pp. 35–42, 1983.

[6] J. G. Carbonell, "The role of user modeling in natural language interface design," Comput. Sci. Tech. Rep. CMU-CS-83-115, Carnegie-Mellon Univ, Pittsburgh, PA, 1983.

[7] J. G. Carbonell and P. J. Hayes, "Recovery strategies for parsing extragrammatical language," Comput. Sci. Tech. Rep. CMU-CS-84-107, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[8] 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*, vol. 21, no. 8, pp. 601–613, 1978.

[9] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human–Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[10] W. B. Croft, "The role of context and adaptation in user interfaces," *Int. J. Man–Machine Studies*, vol. 21, pp. 283–292, 1984.

[11] E. A. Edmonds, "Adaptive man-computer interfaces," in *Computing Skills and the User Interface*, M. J. Coombs and J. L. Alty, Eds. London: Academic Press, 1981, pp. 389–426.

[12] T. Finin and D. Drager, "GUMS: A general user modeling system," Univ. Pennsylvania School of Eng. and Appl. Sci. Tech. Rep. MS-CIS-86-35, May 1986.

[13] G. Fischer, A. Lemke, and T. Schwab, "Knowledge-based help systems," in *Proc. CHI 85 Human Factors in Computing Sys.*, 1985, pp. 161–167, 1985.

[14] 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 Tech. Rep. PFTR-1016-76-8/30, Washington, DC, 1976.

[15] E. E. Goldin and B. Hayes-Roth, "Individual differences in planning processes," Office of Naval Research Note #N-1488-ONR, 1980.

[16] S. Greenberg and I. H. Witten, "Adaptive personalized interfaces —A question of viability," *Behavior and Inform. Tech.*, vol 4, no. 1, pp. 31–45, 1985.

[17] H. P. Grice, "Logic and conversation," in *Syntax and Semantics*, P. Cole and J. Morgan, Ed. New York: Academic Press, 1975, pp. 41–58.

[18] L. Kleinrock, "Distributed systems," *Commun. ACM*, vol. 28, no. 11, pp. 1200–1213, 1985.
[19] J. F. Lehman and J. G. Carbonell, *Learning the User's Language: A Step Towards Automated Creation of User Models*, Carnegie-Mellon University, Pittsburgh, PA, 1987.
[20] B. Liskov and S. Zilles, "Specification techniques for data abstractions," *IEEE Trans. Software Eng.*, vol. SE-1, no. 1, pp. 7–19, 1975.
[21] H. S. Maskery, "Adaptive interfaces for naive users—an experimental study," *Interact '84, Proc. IFIP Conf.*, 1984, pp. 343–350.
[22] M. V. Mason and R. C. Thomas, "Experimental adaptive interface," *Information Technology: Research, Design, Applications*, vol. 3, no. 3, pp. 162–167, 1984.
[23] M. Matz, "Towards a process model for high school algebra errors," in *Intelligent Tutoring Systems*, D. Sleeman and J. S. Brown, Eds. New York: Academic Press, 1982.
[24] J. Meads, "Report on the SIGCHI workshop on planning for user interface standards," *SIGCHI Bull.*, vol. 17, no. 2, pp. 11–16, 1985.
[25] I. Monarch and J. Carbonell, "CoalSORT: A knowledge-based interface," *IEEE Expert*, vol. 2, no. 1, pp. 39–53, 1987.
[26] K. Morik and C. Rollinger, "The real estate agent—Modeling users by uncertain reasoning," *The AI Magazine*, pp. 44–52, Summer 1985.
[27] D. V. Morland, "The evolution of software architecture," *Datamation*, pp. 123–132, Feb. 1985.
[28] A. Morse, "Some principles for the effective display of data," *Commun. ACM*, pp. 94–101, 1979.
[29] H. Mozeico, "A human/computer interface to accommodate user learning stages," *Commun. ACM*, vol. 25, no. 2, pp. 100–104, 1982.
[30] D. A. Norman, "Some observations on mental models," in *Mental Models*, D. Gentner and A. L. Stevens, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983 pp. 7–14.
[31] ____, "Design rules based upon analyses of human error," *Commun. ACM*, vol. 26, pp. 254–258, 1983.
[32] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, pp. 1053–1058, 1972.
[33] ____, "Use of abstract interfaces in the development of software for embedded systems," Naval Res. Lab. Rep. 8047, Washington, DC, 1977.
[34] F. T. Passini and W. T. Norman, "A universal conception of personality structure," *J. Personality Soc. Psych.*, vol. 4, pp. 44–49, 1966.
[35] M. E. Pollack, "Information sought and information provided: An empirical study of user/expert dialogues," in *Proc. CHI 85 Human Factors in Computing Sys.* 1985, pp. 155–159.
[36] K. M. Potosnak, "Choice of interface modes by empirical groupings of computer users," in *Proc. IFIPS*, 1984, pp. 27–32.
[37] R. Reichman-Adar, "Extended person-machine interface," *Artificial Intell.* vol. 22, pp. 157–218, 1984.
[38] ____, "Communication paradigms for a window system," in *User Centered System Design: New Perspectives in Human-Computer Interaction*, D. A. Norman and D. W. Draper, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, 1986, pp. 285–313.
[39] E. Rich, "User modeling via stereotypes," *Cognitive Sci.*, vol. 3, pp. 329–354, 1979.
[40] E. Rissland, "Ingredients of intelligent user interfaces," *Int. J. Man–Machine Studies*, vol. 21, pp. 377–388, 1984.
[41] I. T. Robertson, "Human information-processing strategies and style," *Behaviour and Inform. Tech.* vol. 4, no. 1, pp. 19–29, 1985.
[42] C. D. Rose, "R & D race tightens for 5th-generation computers," *Electron.*, pp. 30–31, Sept. 1985.
[43] W. B. Rouse, "Human-computer interaction in the control of dynamic systems," *Computing Surveys*, vol. 13, pp. 71–100, 1981.
[44] W. Sherman, "SAUCI: Self-adaptive user-computer interfaces," Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 1986.
[45] S. L. Smith and J. N. Mosier, "Design guidelines for the user-system interface software," Mitre Corp. Tech. Rep. ESD-TR-84-190, Bedford, MA, 1984.
[46] G. C. VanDerVeer, M. J. Tauber, Y. Waern, and B. VanMuylwijk, On the interaction between system and user characteristics," *Behaviour and Inform. Tech.* vol. 4, no. 4, pp. 289–308, 1985.
[47] W. Wahlster, and A. Kobsa, "Dialogue-based user models," *Proc. IEEE*, vol. 74, no. 7, pp. 948–960, 1986.
[48] R. Wilensky, Y. Arens, and D. Chin, "Talking to Unix in English: An overview of UC," *Commun. ACM*, vol. 27, pp. 574–593, 1984.
[49] E. Yallow, "Individual differences in learning from verbal and figural materials," School of Educ. Stanford Univ., Aptitudes Research Project Tech. Rep. No. 13, Palo Alto, CA, 1980.