

1989 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS

CONFERENCE PROCEEDINGS

November 14-17, 1989
Hyatt Regency, Cambridge
Cambridge, Massachusetts

89CH2809-2



VOLUME III OF III

THE SOFTWARE ENGINEERING OF ADAPTIVE HUMAN-COMPUTER INTERFACES

A. F. Norcio

Information Systems Management Department
University of Maryland
Catonsville, Maryland 21228

This paper presents a conceptual research framework for studying and designing adaptive human-computer interfaces that modify themselves dynamically with respect to the current user and the current context. This framework is based upon the software engineering principle for constructing an abstract specification that incorporates all the necessary information about the entity. The goal of this approach is to implement an abstract interface that is composed of an abstract user, an abstract system, and an abstract application. At this point in time the work focuses exclusively on designing the abstract user. The abstract user is seen as an interaction between a static production model of the goals necessary to accomplish a specific task and a dynamic production model of a user's goal hierarchy for accomplishing that task. This framework suggests that the task model necessarily follows from the specifics of the task domain. Further, the user's goal model must be constructed by monitoring the user's actions, deducing the goals, and comparing the user's goal hierarchy with the set of goals that the task model requires.

The goal of this research is to provide a framework for constructing adaptive human-computer interfaces by employing software engineering design methodologies. The traditional model for the user interface environment is depicted in Figure 1. This model views the system as the dominant concern. It does not consider the interface as part of the system. And worse, the user is rarely considered.



Figure 1

There are at least two major flaws in this approach. They are:

- 1) the user is not viewed as part of the overall system; and
- 2) the interface is typically an afterthought.

This approach with these inherent flows frequently results in human-computer interfaces that are focused almost entirely on system efficiency and give little or no thought to user efficiency. Moreover, the vigor and intensity that is employed in the design of the interface is rarely equaled in the design of the interface.

This situation can result in systems that are excessively complicated and difficult for the average user. Kleinrock warns that users may become "hopelessly lost" in future complex systems if interfaces are not designed with some level of formality as the underlying system through which they operate [3].

Consequently, this research program suggests that a new human-computer interface model may be more appropriate for complex systems of the future. This model, which is depicted in Figure 2, views the user, the system, and the application as a single unified environment in which all three components are linked with each other through an abstract interface [6, 7].

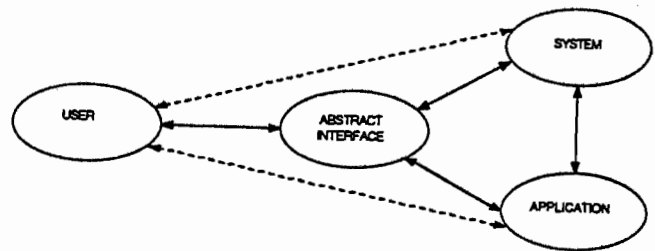


Figure 2

The abstract interface, utilizing the information hiding principle, contains the "secrets" of the user, the system, and the application [5]. In other words, the abstract interface is the real system and the real application as far as the user is concerned. Likewise, the abstract interface is the real user and the real application from the system's perspective; and it is also the real application from the user's and system's view.

The underlying structure of the abstract interface is depicted in Figure 3. The approach used in this project decomposes the abstract interface into three modules. They are:

- 1) the abstract user module,
- 2) the abstract system module, and
- 3) the abstract application module.

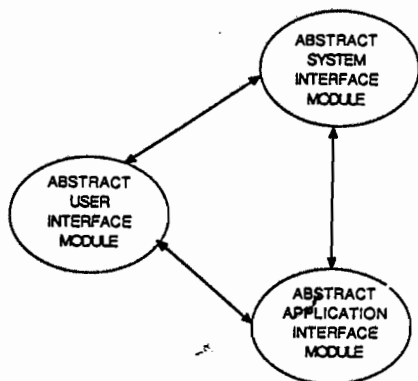


Figure 3

At this point in time, no effort is being spent on the abstract system and application modules. However, a good example of an abstract system module is the UNIX kernel [2]. This module makes the hardware system on which UNIX is operating totally transparent to the user. That is, from the user's perspective, the UNIX kernel makes the specifics of the actual hardware configuration completely inconsequential. This module makes all hardware and system conventions the same for all UNIX users.

Good examples of an abstract application module can be found in several data base interface systems that are currently available. Several of these packages are able to communicate with a variety of data base software environments. However, these interfaces allow users to use the same command and control structures across these different environments.

Again, this software makes the specifics of the actual data base command language inconsequential. These interfaces make all their associated data base packages the same for all the respective users.

ABSTRACT USER

The focus of this project is concentrating on developing a research framework for constructing and studying the abstract user module. The first step in this process is defining the abstract user and hiding this information in the abstract user module.

The definition of a user requires a user model that reflects the typical user. The current structural model of a typical user is presented in Figure 4.

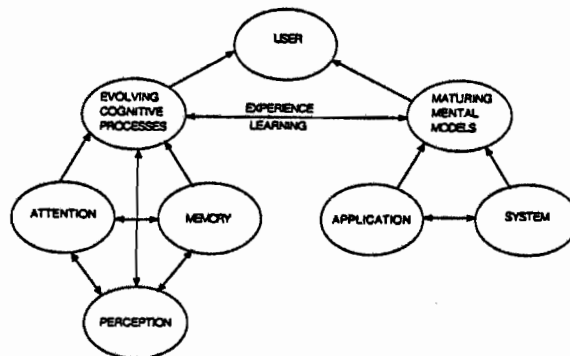


Figure 4

This model assumes that all users are composed of two fundamental cognitive processes. They are Evolving Cognitive Processes (ECP) and Maturing Mental Models (MMM).

A number of cognitive processes could legitimately be included in ECP. This project has decided to include only three subprocesses; namely, Attention, Memory, and Perception. Of these three, attention is the most difficult to examine. Memory and perception, however, have a rich experimental literature upon which to draw [e.g. 1, 4].

Consequently, the initial set of experiments focus on memory and, to a lesser extent, perception and attention. The interaction between these three subprocesses form an individual user's level of cognitive processes.

Likewise, a number of mental models could be included under Maturing Mental Models. Since the goal of this work is to provide a framework for adaptive human-computer interface, only two models are included. They are the user's mental model of the application and the user's mental model of the system.

There are several assumptions that are implicit in this model. They include:

- 1) there is an interaction between the ECP subprocesses,
- 2) there is an interaction between the mental models that comprise the MMM,
- 3) there is an interaction between the ECP and the MMM and that this interaction can be labeled experience/learning,

- 4) the level of a user's experience/learning is manifested by the pattern of commands and requests presented to the interface,
- 5) the command pattern, which inherently results from the interactions between all the subprocesses and submodels, explicitly reflects the user's goals.

The purpose of the experimentation that is currently underway is to define parameters for memory and perception as well as to define inferential rules that can deduce goals and experiences from command sequences.

ABSTRACT USER INTERFACE MODULE

The ultimate goal of this effort is to implement an Abstract User Interface Module. The purpose of this module is to infer categorical information about the user's cognitive memory and perception structures, current mental models of the application and the system, and the user's ultimate goals. The model should be able to deduce and alter these inferences dynamically. As a result of these inferences, the abstract user interface module should adjust system responses to the user, modify the presentation and organization of information, and detect user difficulties in satisfying the inferred goals. In other words, the abstract user module should provide support facilities that are customized for each individual user.

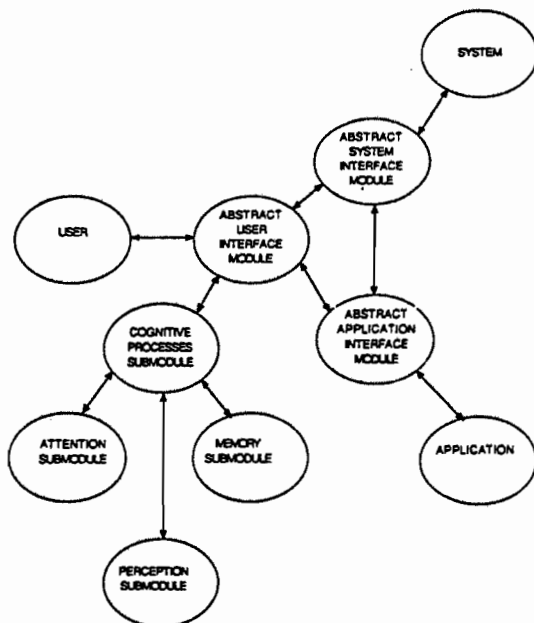


Figure 5

The overall adaptive environment is presented in Figure 5. This model assumes that the specific user, system, and application are essentially irrelevant to an adaptive human-computer interface if the software engineering principles of abstract interfaces and information hiding are used in their design and implementation.

REFERENCES

1. Chase, W. G. and Simon, H. A., "Perception in Chess," Cognitive Psychology 4, pp. 55-81 (1973).
2. Kerningham, B. W. and Pike, R., The UNIX Programming Environment, Prentice-Hall, Inc., Englewood Cliffs, NJ (1984).
3. Kleinrock, L., "Distributed Systems," Communications of the ACM 28(11), pp. 1200-1213 (1985).
4. Miller, G. A., "The magical number seven, plus or minus two: Some limits on our capacity for processing information." Psychological Review 63(2), pp. 81-96 (1956).
5. Parnas, D. L., "On the criteria to be used in decomposing systems into modules," Communications of the ACM 15, pp. 1053-1058 (1972).
6. Parnas, D. L. and Clements, P. C., "A rational design process: How and why to fake it," IEEE Transactions on Software Engineering SE-12(2), pp. 251-257 (1986).
7. Parnas, D. L., Clements, P. C., and Weiss, D. M., "The modular structure of complex systems," IEEE Transactions on Software Engineering SE-11(3), pp. 259-266 (1985).