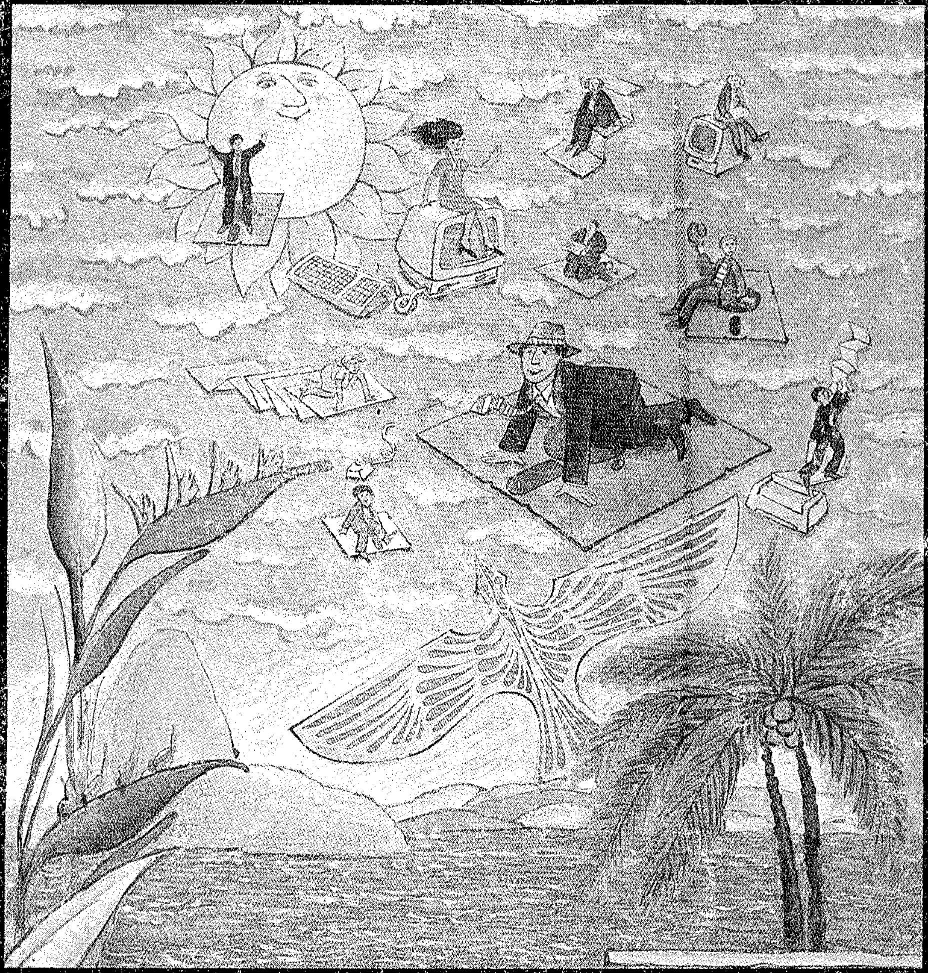
1ST INTERNATIONAL INFORMATICS CONGRESS OF RIO DE JANEIRO



THE IMPACT OF USING INFORMATION TECHNOLOGY

PROCEEDINGS

BRAZIL, 22 to 26, August 1988

France

Association Française des Informaticiens
France

A.I.C.A.
ASSOCIAZIONE ITALIANA PER L'INFORMATICA ED IL CALCOLO AUTOMATICO
Italy

A.P.I.
ASSOCIAÇÃO PORTUGUESA DE INFORMÁTICA
Portugal

The British Computer Society
England

DPMA
Data Processing Management Association
USA

GESELLSCHAFT FÜR INFORMATIK E.V.
Germany

THE INFORMATION PROCESSING SOCIETY OF JAPAN
Japan

SUCESU
sociedade dos usuários de computadores e equipamentos subsidiários
Brazil

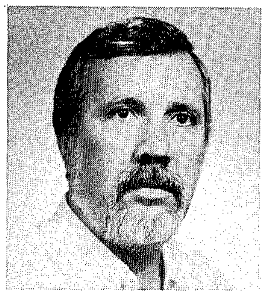
1st INTERNATIONAL INFORMATICS CONGRESS OF RIO DE JANEIRO

PROCEEDINGS

BRAZIL, 22 TO 26, AUGUST 1988

NEW TRENDS AND ISSUES IN COMPUTER SYSTEM TECHNOLOGIES FOR USER'S INTERFACE*

CARLOS J. P. LUCENA
DEPARTAMENTO DE INFORMÁTICA
PONTIFÍCIA UNIVERSIDADE CATÓLICA
RIO DE JANEIRO



Doctor in Computer Science at the University of California, Los Angeles; Vice-Rector and Professor of Computer Science at the Pontificia Universidade Católica do Rio de Janeiro; Guggenheim fellow in Computer Service; Chairman of the area of Theorical Foundations of Information Systems, IFIP World Conference, Japan, 1980; author of 3 books and 50 technical papers.

*This work was partially supported by SID Informática as part of the ESTRA project.

1. INTRODUCTION

The new user of an interactive system must learn both the functionality of the system and the procedures to invoke its functions. Often he can transfer his knowledge of the functionality from one system to another with more or less success; rarely can he do the same for the form of the interaction process.

From the designer's point of view, creating good user interfaces is a difficult process that requires iterations of design and evaluation. The work to provide assistance to the interface designer falls broadly into three areas. One area involves simulating systems before they are built to improve the interface design through observation and modification of a model of the system. A second area describes formally the structure of human-computer interaction. A third area develops tools for producing user interfaces.

Developing a user interfaces takes a lot of time beucause of the following features of modern input techniques: (1) the interface needs to respond to many modern input techniques which are menu-based, form-based, mouse-driven and touch screen-driven; (2) we would like the interface to adapt its behaviour to the level of expertise of the user (e.g. experts and novices) and (3) features such as aliasing, command completion, command combination, history mechanism and I/O redirection make the task more difficult.

Techniques from artificial intelligence such as plan recognition, plan generation and user-modelling are of great interest to the design of advice-given systems, in particular when combined with natural language processing techniques.

This paper stress the importance of designing superior human-machine interfaces to the modern user by studying up-to-date design strategies and techniques that try to achieve that goal. Various techniques will be exemplifiyed through research results produced in various institutions worldwide. Our current research in the area of engineering environments for users interface design and implementation which takes place in the context of the ESTRA project is highlighted at the end of the paper.

2. USER'S INTERFACES AS THE USER SEES IT

The general problem of making computers easier to use is a major enterprise of obvious importance. A User's Interface (UI) is any computer software that has as its primary function the task of providing the user with information that will assist him in the use of some other software system. We here assume that the single most important criterion by which a UI system should be judged is the degree to which it facilitates the accomplishement of a particular task by a user who did not previously know how to accomplish this task. What, then, should a UI actually look like? Indeed, is it even possible for a UI to be of more assistance than a paper manual, which certainly presents the simples mechanism imaginable for most users? How should a UI be designed and engineered?

People have been building UI's almost as long as they have been building software systems of any kind. In general, the UI's have been an aftertought, quickly constructed and only marginally integrated into the larger system. However, with the increasing availability of computer technology to the general public, there is a growing awareness of the importance of UI's. This has led to an increased effort in the research and development in UI engineering. Sondeheimer (1) and Borenstein (2) have proposed seven dimensions along which a UI as the user sees it may vary. The taxonomy assumes in arbitrarily fast computer with otherwise ideal hardware and sufficient memory to easily retain all relevant contextual information. In other words the interest is focused on design and not implementation (both hardware and software). The dimensions are: acess issues (three aspects), presentation issues (three aspects) and integration.

Borenstein claims that there are three major issues in the user's access to a help system: whose initiative first formulates the UI activity, how the user may request further help and how complex the help request language is.

In classifying UI systems according to acess initiative, such systems are placed on continuum between systems in which human users have the sole initiative and systems in which the computer more often takes the initiative than the human, but systems with somewhat mixed initiative are common.

The number of access mechanisms implemented or even proposed is actually very small. The six most popular mechanisms are: key word help, menu help, contextually invoked help, graphically invoked help, natural language help requests and spoken help requests.

The third aspect of the access issue is access complexity. Virtually all of the mechanims described above may be implemented well or poorly from the users perspective. The difference can be overwhelming in terms of its effect on the system's effectiveness. Syntax is important, as well as the branching factor in menu systems (too many choices or too few choices). In graphics-based systems issues of what icons should look like and how selection with a mouse should be achivied are still in general unresolved.

On-line help varies greatly not only in how it is accessed but in how it is presented as well. We explore now the three principal dimensions along which the presentation of help information may vary: presentation methods, presentation source and text quality.

Video screen technology has allowed more modern help systems to use multiple windows. In such a system the user can preserve his context at the bottom of the screen, for example, while scrolling through help texts on the top of his screen. The penalty here, of course, is that the avaiable screen size for each of these activities is only half of the screen size to which the user is otherwise accustomed. The future holds the promise of new technologies that might be useful for the presentation of help information. Synthesized speech might provide help without sacrificing any of the screen territory.

Besides the question of how the help information is to be presented, there is the question of where it is to come from. The fundamental issue here is that the text can either be retrieved verbatim from some data structure that contains it, or it may be generated on-the-fly by some natural language composition mechanisms acting on an underlying knowledge representation. It may be useful to use language generation facilities to dinamically generate examples or to customize explanations to a specific context in which the user is having difficulties.

The final dimension along which help presentation varies is that of text quality. The literature on tex readability is enourmous and nearly all of it applies to help texts. What this suggests primarily is that such texts should be designed by a specialist in the field of linguistic and cognitive aspects of tex comprehension. Improvements in this area will undoubtedly be closely linked to developments in the field of user-modelling.

Finally, perhaps the most important aspect of a UI system as a whole is its level of integration.

Many computer systems provide several on-line help mechanisms, each with its own database, operating completely independently. The virtues of integration are obvious: by providing uniform access to help you eliminate confusion for the user and make it easier for him to stay in context. By making various mechanisms access a single help database you make it easier for the user to try all of the mechanisms you provide in his attempt to learn what he needs to know.

The following systems are representative of the diversity of advanced help systems that have been previously implemented: Emac (3), Wizard (4), Star (5), Berkeley Unix Help System (6), Unix Consultant (15), Browse (7) and Interlisp Dwin (8).

3. DESIGN PRINCIPLES AND TOOLS FOR THE CONSTRUCTION OF UI'S

Man machine communication is implemented through a variety of physical devices whose characteristis are extremely diverse. A UI implementation would be an extremely difficult task if the developer needed to know the control details of all devices. To overcome this problem a number of graphic primitives have been proposed to provide an abstract view of these devices by hiding their internal details. There exists today a number of graphical packages that isolate the application package from the software needed to control the physical devices.

Within the GKS standard, for instance, high level notions such as workstation, logical inputs, logical outputs can be expressed. The logical inputs try to capture the typical tasks that take place during man-machine interaction. They are: identify a position, identify a sequence of positions, quantify a value, choose a value from within a set, point to an object and produce a chain of characters. It is obvious that between this level of abstraction and the level perceived by the user (as in the previous section) there exists a gap. The classical software engineering procedure consists of developing a set of tools on top of the primitives to allow the design and implementation at a level of abstraction closer to the notions perceived by the final user. These tools support abstractions such as: menus, commands, input forms, objects with a defined graphical representation, numerical values etc. The tools constitute a software engineering environment that allows the interface designer to create and edit the external representation of the objects the user will see during interaction. The environment deals not only with with the static aspects since they also need to support the dynamic aspects of the interaction, such as the echo and the feedback associated with a user's action.

Since there are many possible ways of expressing the dynamic aspects that arrive during an arbitrary interaction with a UI it is convenient to define classes of interaction styles. When tools are available to support the use of a style definition, the tools that support the implementation of complex interactive tasks can be instanciated with default values defined by the style. Styles are defined by UI designers based on previous work on user modelling.

There are essentially three types of engineering environments for UI design and implementation. The first possibility is the development of new programming languages suited to interface design. IFS (9) is an example of this type of work. This approach has serious limitations because the language commands that embed the graphic primitives lack the necessary high level of abstraction refered to at the begining of the section. A second approach in the attention of abstraction libraries that can be accessed from a program in the special purpose language. The third and already established approach is the use of an engineering environment dedicatet to interface design and implementation.

Many on going research and development projects have demonstrated the viability of generating a UI from a specification of the user-computer dialogue. The dialogue can be formalized through the use of Petri Nets, augmented transition networks and formal grammars (among other techniques). When deduction is required a knowledge base is incorporated into the system to store knowledge about the user, the application and the use of the various features of the UI.

The development process of an interface when this type of environment, is available starts from a dialogue specification, which is transformed to produce an interface prototype. The prototype is then handled interactively by the designer who tunes it into a suitable interface software. Note that this procedure implies that the application software is developed in paralel in a way that guarantees its future integration to the interface.

Much like software engineering environments in general, today's environments for interface construction fall into two broad categories. In the first case the tools available are based on well-defined algorithms and the center of the system is a DBMS (e.g. The Aide System (10) In the second case the system is capable of making deductions, tries to model the user's cognitive behaviour and has a knowledge based system (at least one) as its central feature (e.g. Planex (11).

Towards an Ideal Engineering Environment for UI's Development

Many research groups in many countries are deeply involved in research leading to an "ideal" environment for UI's development. In Brazil, one major project in this area is the ESTRA project. ESTRA is a cooperative project led by the research division of SID Informatica in cooperation with ten universities in the country including my own. I was responsible for the original statement of goals of the project and for the formulation of the strategy for managing it as a cooperative enterprise (12). ESTRA (stands for Estação de Trabalho, which means advanced workstation in portuguese) is meant to be a dedicated workstation (both hardware and software) specialized in the design and implementation of integrated UI's for a given set of applications. The workstation is to be used either in a stand alone mode or as part of a network set up for cooperative design and development.

Since its first definition in late 1985, ESTRA is meant to encompass both the characteristics of a data base centered environment and the features of an environment that supports deductive mechanisms such as planning and explanation. ESTRA is a five year project and so far a number of interesting intermediate results have been published and demonstrated. Our research group at PUC/RJ, in particular, has been interested in the development of the tutors for a variety of applications, has generated plan based interfaces for particular applications and is in the process of generalizing this experiment to produce an appropriate environment to assist in the creation of plan-based interfaces. Work with object-based implementations has been providing good insights on how to organize the software base for the applications we want to see integrated and on how to deal with cooperative development of an interface in an object-based environment. Besides integrating these results with those of the other groups within the project, such as natural language processing, image processing and formal software specification, among others, we have been aiming at some new goals within our module of the ESTRA project. In particular we want to explore the potentiality of the plan generation approach in the presence of the issue of concurrency in users interfaces. There are three basic types of concurrency at the user interface (13) that presents problems to the plan generation/recognition approach:

1. Concurrent output (exemplifyed by the simultaneous updating of several windows on a single display)
2. Concurrent input (exemplifyed by the situation in which a workstation that has both a mouse and a keyboard also has an operating system that is able to accept input from both simultaneously)

3. Concurrent dialogues which are a form of concurrency whereby the user may supply input to several (possibly related) user interfaces simultaneously.

The issues affect not only the complexity of the plan generation strategies but also induces new problems in the area of designing an appropriate operating system for ESTRA (14).

Along the year of 88 results from various participants of ESTRA project will be consolidated in a prototype ESTRA workstation that will extensively evaluated by the participant institutions.

REFERENCES

(1) Sondeheimer, N. and Relles, N.; "Human Factors for User Assistance in Interactive Computing Systems: An Introduction", IEEE Transactions on Systems, Man and Cybernetics, 12 (2), March-April 1982.

(2) Borenstein, N.S.; "The Design and Evaluation of On-line Help Systems", Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, 1985.

(3) Gosling, J.; "UNIX Emacs Manual", 1983.

(4) Finin, T.W.; "Providing Help and Advice in Task-Oriented Systems", in IUCAI 83 Proceedings, 1983.

(5) Smith, D.C. et alii; "The Star User Interface: An Overview", in National Computer Conference Proceedings, AFIPS, 1982.

(6) Kunze, J.; "The Berkeley UNIX Help Systems", 1984.

(7) Bramwell, B.; "Browse: An On-line Manual and System Without an Acronym", SIGDOC Newsletter, 1984.

(8) Teitelman, W.; "Interlisp Reference Manual", Xerox Palo Alto Research Center.

(9) Vo, K.P.; "IFS – A Tool to Build Integrated, Interactive Application Software", AT & T Technical Journal, Vol. 64, nº 9, 1985.

(10) Gordon, R.F.; "Application Interface Development Environment", Research Report, IBM Research Division, 1985.

(11) Michard, A.; "Reconnaissaissance et Generation de Plans d'action: Application a la Realization de Systemes Auto Explicatifs", in Proceedings Cognitiva 85, Paris, 1985.

(12) Lucena, C.J.; "Um Foco para o Esforço de Pesquisa da SID: Uma Estação de Desenvolvimento para a Síntese de Aplicações Integradas", Memorando de Pesquisa 86/001, SID Informática, 1986.

(13) Hill, R.D.; "Supporting Concurrency, Communication and Synchronization, in Human-Computer Interaction – The Sassafras UIMS", ACM Transaction Graphics, Vol. 5, nº 3, 1986.

(14) Balzer, R.M.; "Living in the Next Generation Operating System", in Information Processing 86, H.J. Kungler (ed.) Elsevier Science Publishers (North Holland), 1986.

(15) Wilensky, R. et al.; "UC: A Progress Report", Technical Rep. UCB/CSD 87/303, Computer Science Division, University of California at Berkeley, 1986.