

**USER INTERFACES FOR
COMPUTER-AIDED ARCHITECTURAL DESIGN**

**A THESIS
SUBMITTED TO THE DEPARTMENT OF
INTERIOR ARCHITECTURE AND ENVIRONMENTAL DESIGN
AND THE INSTITUTE OF FINE ARTS
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF FINE ARTS**

Aygün Kulaksız
tarafından bağışlanmıştır.

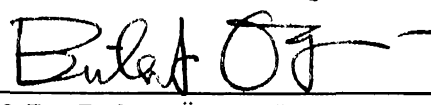
**By
AYGÜN KULAKSIZ**

February, 1993

NA
2728
.K85
1992

B762

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Fine Arts.



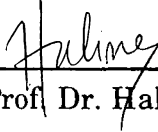
Prof. Dr. Bülent Özgüç (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Fine Arts.



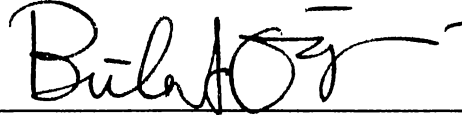
Prof. Dr. Mustafa Pultar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Fine Arts.



Assist. Prof. Dr. Halime Demirkan

Approved by the Institute of Fine Arts.



Prof. Dr. Bülent Özgüç, Director of the Institute of Fine Arts

ABSTRACT

USER INTERFACES FOR COMPUTER-AIDED ARCHITECTURAL DESIGN

Aygün Kulaksız

M. F. A. in

Interior Architecture and Environmental Design

Supervisor: Prof. Dr. Bülent Özgüç

February, 1993

The rapidly developing technology of the twentieth century has transformed the general use of computers into a specific, convenient, and necessary tool for professionals. As in each profession, they are also used by architects. But, architects have some problems with the properties of user-computer interface that inherit from the times when computers were only used by computer professionals. Considering the architects professional needs and expectations, this thesis intends to avoid the unsatisfying results of this poor dialogue. After mentioning the development of human-computer interaction, the specific problems that a new user may face and the characteristics of a well designed interface are described. Although there are much more primitive action units performed by the user, the essential ones such as interaction tasks, the complementaries like controlling tasks that may be preferred by architects are examined. Different types of interaction techniques which respond to the various kinds of requirements of these tasks are explained, by identifying their advantages and disadvantages. In order to establish the architects' intended goals, some formal specifications, standards and prototypes that are required by the increasing needs for communication, the access of information technology and the rising involvement of architects into the computer-aided technology, are identified. Gradually the evaluation of the interface is stated as a guideline both for the architect who wants to use a software and the computer programmer who wants to write a software for the architects.

Keywords: computer-aided architectural design, user interface, human-computer interaction, human-machine interface.

ÖZET

BİLGİSAYAR DESTEKLİ MİMARİ TASARIMDA KULLANICI ARABİRİMLERİ

Aygün Kulaksız

İç Mimarlık ve Çevre Tasarımı Bölümü

Yüksek Lisans

Tez Yöneticisi: Prof. Dr. Bülent Özgüç

Şubat, 1993

Günümüzün hızlı gelişen teknolojisi bilgisayarı genel amaçlı kullanımdan çıkarıp meslekler için özgün, münasip ve gerekli araçlar haline çevirmiştir. Bilgisayarlar birçok meslekte olduğu gibi mimarlar tarafından da kullanılmıştır. Fakat mimarların kullanıcı-bilgisayar arabirimleri ile ilgili bazı sorunları vardır. Bunlar bilgisayarların, sadece bilgisayar uzmanları tarafından kullanıldığı zamandan miras kalmıştır. Bu tezin amacı, mimarların mesleki ihtiyaçlarını ve beklentilerini dikkate alarak bu zayıf diyalogun başarısız sonuçlarını incelemektir. İnsan-bilgisayar etkileşiminin gelişimi incelendikten sonra, yeni kullanıcıların karşılaşılabileceği özgün sorunlar ve iyi tanımlanmış bir etkileşimin özellikleri temellendirilmiştir. Kullanıcıların çok çeşitli temel çalışma birimleri olduğu için, daha çok mimarların tercih ettiği ana etkileşim birimleri ve bunların tamamlayıcı kontrol mekanizmaları incelenmiştir. Bu etkileşim birimlerinin çeşitli gerekliliklerini yerine getirecek farklı etkileşim teknikleri avantaj ve dezavantajları açıklanarak tanımlanmıştır. Mimarın beklenen hedeflerini karşılamak amacıyla, artan iletişim ihtiyacı, bilgi erişim teknolojisi ve bilgisayar destekli teknolojiye mimarların katılımının artışının getirdiği bazı resmi şartlar, standartlar ve prototipler incelenmiştir. Sonuç olarak hem bir mimari bilgisayar yazılımı kullanmak isteyen mimara, hem de mimarlar için bilgisayar yazılımı hazırlayacak olan programcılara rehber olması amacıyla, bilgisayar insan etkileşimi sonuçları değerlendirilmiştir.

Anahtar Kelimeler: bilgisayar destekli mimari tasarım, kullanıcı arabirimi, insan-bilgisayar etkileşimi, insan-makine arabirimi.

ACKNOWLEDGEMENTS

I have thoroughly enjoyed the professional manner in preparing this master's thesis. I am indebted to many for their ideas and assistance. My primary obligation is to my advisor Prof. Dr. Bülent Özgüç who helped make this study possible with his precious knowledge, critique, time and support.

This thesis reached its final printed form with the tireless and encouraging efforts of my sister Ind. Eng. Aycan Kulaksız during each step of the project. I give my special thanks for her contributions.

This work has benefited significantly from comments and suggestions received from various academics. I deeply appreciate the kindness and generosity of Assist. Prof. Dr. Halime Demirkan and Com. Eng. Mesut Göktepe for hours of discussions and for supplying me with valuable information.

Most of all, special encouragement was given to me during the writing of this thesis by my family, Melda, Aysun, and Ayşen Kulaksız. My indebtedness to an understanding family is hereby reaffirmed.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ÖZET	iv
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF TABLES	viii
LIST OF FIGURES	ix
1. INTRODUCTION	1
2. HUMAN - COMPUTER INTERACTION	10
2.1. Development of Human Computer Interaction.....	11
2.2. Typical Problems Faced By New Users.....	16
2.3. Characteristics of a well Designed Human - Computer Interface.....	18
2.3.1. Human Side of the Interface.....	23
2.3.1.1. User Interface Requirements.....	24
2.3.1.2. Accessibility.....	25
2.3.1.3. Starting and Terminating Sessions.....	26
2.3.1.4. Training and User Aids.....	26
2.3.1.5. Vocabulary.....	28
2.3.2. System Side of the Interface.....	29
2.3.2.1. Functionality and Visual Interface.....	30
2.3.2.2. System Dynamics and Response Time.....	31
2.3.2.3. Work-session Interrupts.....	32
2.3.2.4. Consistency and Compatibility in Interaction.....	33
2.3.2.5. Visibility and Simplicity.....	34
2.3.2.6. Data Organization.....	34
2.3.2.7. Dialogues.....	35

3. GRAPHICAL INTERACTION AND CONTROLLING TASKS	37
3.1. Looking For Guidance.....	38
3.2. Scope of the Problem.....	39
3.3. Measures of Ergonomic Quality.....	40
3.3.1. Primary Criteria.....	40
3.3.2. Secondary Criteria.....	41
3.4. Interaction Tasks.....	42
3.4.1. Select.....	43
3.4.2. Position.....	44
3.4.3. Orient.....	45
3.4.4. Path.....	45
3.4.5. Quantify.....	46
3.4.6. Text.....	47
3.5. Controlling Tasks.....	48
3.5.1. Stretch.....	48
3.5.2. Sketch.....	49
3.5.3. Manipulate.....	51
3.5.4. Shape.....	53
4. INTERACTION TECHNIQUES	54
4.1. Command Language Interface.....	56
4.2. Natural Language Interface.....	59
4.3. Menu-Driven Interface.....	62
4.4. Iconic Interface.....	67
4.5. Graphical Interface.....	71
4.6. Form-Filling Interface.....	74
4.7. Window-Oriented Interface.....	76
4.8. Direct Manipulation.....	80
4.9. Speech Communication.....	83
4.10. Multi-Media Communication.....	86
4.11. Virtual Reality.....	89
5. TOOLS FOR ARCHITECTURAL USER INTERFACE DESIGN	93
5.1. Formal Specification of the Architectural User Interface Design.....	94
5.2. Need for Prototypes of the Architectural User Interface Design.....	97
5.3. Standardization of the Architectural User Interface Design.....	100
6. CONCLUSION: EVALUATION OF THE INTERFACE FOR CAAD	103
REFERENCES	107

LIST OF TABLES

Table	Page
Table 2.1. Developments through seven generations.....	12
Table 2.2. Typical problems faced by new users.....	17
Table 2.3. Typical characteristics of an exploratory environment.....	18
Table 2.4. To be successful, design for interaction between user, task and the system must be based upon these five fundamental features.....	23
Table 2.5. Dialogue design recommendations.....	36
Table 4.1. The items used in graphic communication workstations.....	74
Table 4.2. Considerations in the development, selection and evaluation of automatic speech recognition system.....	86

LIST OF FIGURES

Figure	Page
Figure 1.1. The man-machine system design process.....	4
Figure 1.2. The expected inversion of the pattern of the design effort over the total brief-design-build-use process.....	5
Figure 2.1. Use of human resources in interface design.....	19
Figure 2.2. Knowledge required in an interaction.....	20
Figure 3.1. Selection techniques.....	43
Figure 3.2. Positioning techniques.....	44
Figure 3.3. Orienting techniques.....	45
Figure 3.4. Quantifying techniques.....	47
Figure 3.5. Text-entry techniques.....	47
Figure 3.6. Typical stretching techniques.....	49
Figure 3.7. Computer recognition of a freehand sketch prepared as an input.....	50
Figure 3.8. Translation of an object.....	51
Figure 3.9. Rotation of an object.....	52
Figure 3.10. Scaling an object.....	52
Figure 3.11. Reflection of an object.....	52
Figure 4.1. A sample of command language interface.....	57
Figure 4.2. A sample of natural language menu.....	61
Figure 4.3. Different types of menu selection applications.....	63
Figure 4.4. The pull-down menu on the Apple Macintosh MacWrite program.....	64
Figure 4.5. The linear sequence of menus on the Xerox Star.....	65
Figure 4.6. The scope of iconic communications.....	67
Figure 4.7. Object-based objects currently found on the Apple Macintosh.....	69
Figure 4.8. An abstract/symbol icon for "utilities".....	69
Figure 4.9. Drawing icons carrying a relational structure similar to the one found in the line/rectangle/rectangular prism relationship.....	69
Figure 4.10. An example of a multidimensional icon which represents a file and its five distinct views.....	71
Figure 4.11. The way the user interacts with a multidimensional icon.....	71

Figure 4.12. A CAD software on the Apple Macintosh.....	72
Figure 4.13. A form fill-in design for a department store.....	75
Figure 4.14. A window-oriented interface from a software of the Apple Macintosh.....	77
Figure 4.15. An Apple Macintosh software that offers direct manipulation.....	81
Figure 4.16. Components of a Media View document.....	88

1. INTRODUCTION

The concept of the machine seems to have dominated architectural design philosophy in the twentieth century. Every architect knows Le Corbusier's slogan of the 1920's: "a house is a machine for living in", and many will also have heard of "the architecture machine" of half a century later (Negroponte, 1970).

Although the "machine for living in" and the "architecture machine" may appear to share a common philosophical heritage, they actually represent very different concepts of the role of the machine in architectural design. For Le Corbusier the machine was a source of aesthetic inspiration; he was concerned primarily with the architectural *product*, which he wanted to look like, feel like, and be constructed like a machine. However, the "architecture machine" relates primarily to the architectural *process*; it is a machine for designing; a computer which might have a human partner, but which might also be a designer in its own right.

Negroponte has "adopted the position that computer-aided architecture had to be treated as an issue of machine intelligence". His ideal was a concept of an architecture machine that "must understand our metaphors, must solicit information on its own, must acquire experiences, must talk to a wide variety of people, must improve over time, and must be intelligent". He wanted to build machines "that can learn, can group, and can fumble" (Negroponte, 1970).

The rapid advance in computer technology transformed the computer into a useful, convenient and necessary tool for a wide variety of users including students, business people, managers, designers and researchers. The style of the existing user interface software inherits properties from the times when computers were used only by computer professionals. The outcome of this is a poor user-computer dialogue and dissatisfying results in meeting the end user's wider job needs and expectations.

The earliest forms of the man-machine communication were numerical machines codes (in the late 1940s). In the 1950's these codes were gradually superseded by primitive computer programming *languages*, but it was not until

the 1960's that high level languages enabled non specialist users to have access to computers. Even the widespread computer applications of the 1970's do not permit much more man-machine *communication* than may be possible through a standardized format of predetermined question-answer interrogation.

Today, there is a growing concern for the usability and user friendliness of computer systems as stated by Moran in the following quotation:

A system does not, alas, terminate at its terminals-users attached. The user is one of the critical components determining whether the system is a whole -the human-computer system- works or not (Moran, 1981).

Both the designer and the machine should track each other's design maneuvers, evoking a rhetoric that cannot be anticipated. The event is circular in as much as the designer-machine unity provokes a dialogue and the dialogue promotes a stronger designer-machine unity. Negroponete stated this progressively intimate association of the two dissimilar species as "symbiosis". In order to have a cooperative interaction between the designer and a machine, the two must be congenial and must share a common language (Negroponte, 1970).

With direct, fluid, and natural man-machine discourse, two former barriers between architects and computing machines would be removed. First, the designers, using computer-aided design hardware, would not have to be specialists. Instead, with simple negotiations, the job would be formulated and executed in the designer's own idiom. As a result, a vibrant stream of ideas could be directly channeled from the designer to the machine and back (Negroponte, 1970).

The second obstruction overcome by such close communion is the potential for re-evaluating the procedures themselves. At first a designer may have only a meager understanding of his specific problem and thus require machine tolerance and compatibility in his search for the consistency among criteria and form and method, between intent and purpose. The progression from visceral to intellectual can be articulated in subsequent provisional statements of detail and moment-to-moment reevaluations of the methods themselves (Negroponte, 1970).

If a system is not tailored to the needs and limitations of users, then the users will face difficulties in using the system and this will cause a decrease in productivity, waste of time, and waste of effort.

An interactive computer program may be intended to enable its user to do a variety of different things -find information, compose and format a document, play a game or explore a virtual world. The user's goals for a given application may be recreational, utilitarian, or some combination of both, but it is only through *engagement* at the level of the interface that those goals can be met. An interface like a play, must represent a comprehensible world comprehensibly. That representation must have qualities which enable a person to become engaged, rationally and emotionally, in its unique context.

One of the major difficulties that hinder progress in this field of man-computer symbiosis is that "cognitive work" is not itself an established, static concept. The role that a computer can play in a problem-solving system will depend on what is known about how to solve the particular problems that the system is designed for. In this respect, the original perspective insight of Lady Lovelace in reference to Babbage's Analytical Engine in the mid-nineteenth century still holds: a computer can do "whatever we know how to order it to perform".

Licklider outlines that "the question is not 'What is the answer?' The question is 'What is the question?' " One of the main aims of man-computer symbiosis is to bring the computing machine effectively into the formulative parts of technical problems. The other main aim is closely related. It is to bring computing machines effectively into processes of thinking that must go in "real time", time that moves too fast to permit using computers in conventional ways (Licklider, 1960).

Drawing with a computer is a little like driving: if the destination and route are planned, the trip will be more pleasant and efficient. Planning might result in the fastest possible trip, or it might leave opportunities for scenic side trips. Likewise, successful computer-aided drawing requires certain amount of preparation; both the objectives and the basic structure of a drawing can be defined, much as the goal and intermediate points of a trip are mapped out. If the schedule allows for side trips, this time can be used for additional drawing or for exploration of alternative designs.

In the user's mind, a computer should be a complement: computers have considerable power for data manipulation, but no creative ability, but the architect has the intuition and experience, which is difficult to build into computer systems. The problem is to match the attributes of the architect with those of the computer system.

Conventionally the man-machine systems designer adopts a procedure based on the identification and specification of functions to be performed in the system, and the subsequent allocation of these functions to man or machine in accordance with relative human and machine abilities (Figure 1.1.).

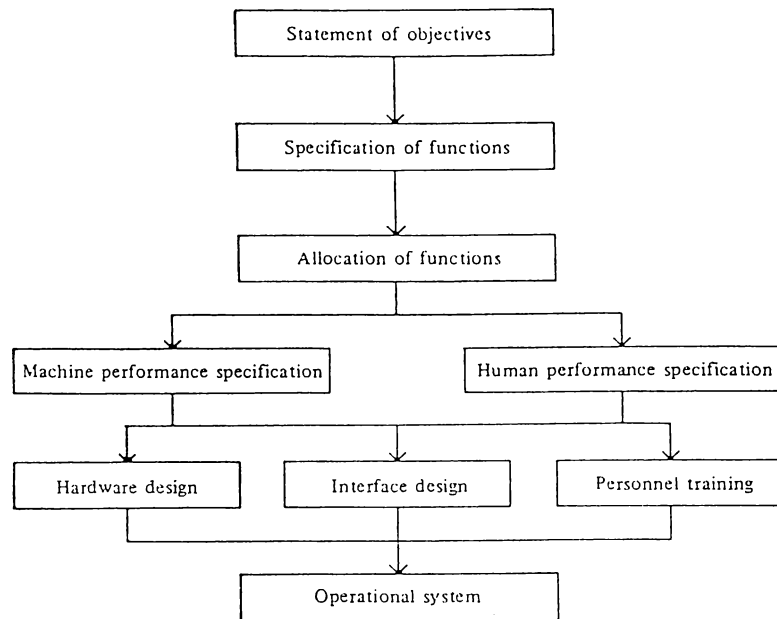


Figure 1.1. The man-machine systems design process (Cross, 1977)

Humans learn particulars and remember generalities, study the specific and act on the general, and in this case the general conflicts with the particular. The problem is therefore twofold: first architects cannot handle large scale problems for they are too complex, second architects ignore small scale problems for they are too particular and individual. Architects do not appear to be well trained to look at the whole urban scene; nor are they apparently skilled at observing the needs of the particular, the family, the individual (Negroponte, 1970).

The operations that fill most of the time allegedly devoted to technical thinking are operations that can be performed more effectively by machines than by men. Severe problems are posed by the fact that these operations have to be performed upon diverse variables and in unforeseen and continually changing sequences. If those problems can be solved in such a way as to create a symbiotic relation between a man and a fast information-retrieval and data-processing machines, however it seems evident that the cooperative interaction would greatly improve the thinking process.

A good drawing program insulates the architect from having to think too much about the organization of the database, and translates instructions into easy to

understand terms that relate to drawing, rather than to database management.

Computers can supplement familiar skills in remarkable ways. But also computer-aided architecture is not without problems; because it is bringing a transformation, demanding not only new skills, but also new promises and principles. They are changing the ways we draw and the ways we use information. These tools have the potential to make the labor of architecture more productive, but, more importantly, they promise to transform the way we design. If used well, they are tools that can add to the creative spark that is so important to architecture.

An overall change in structure of the total design-build-use process which was forecast in the Department of the Environment report was the inversion of the present pattern of effort applied over the process (Figure 1.2.).

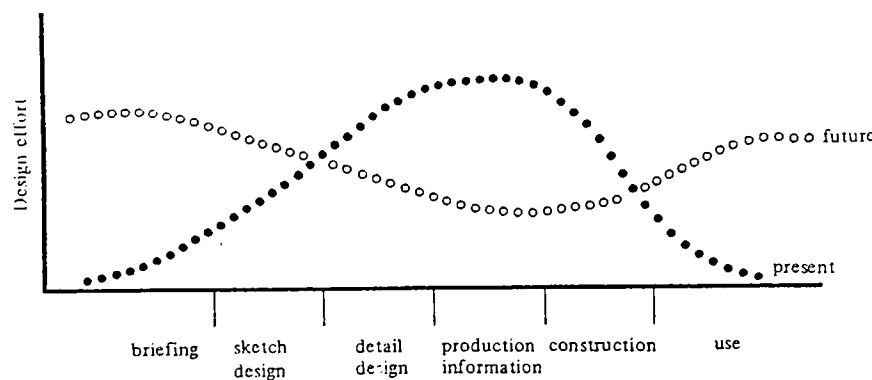


Figure 1.2. The expected inversion of the pattern of design effort over the total brief -design-build-use process (Cross, 1977)

Currently the process is organized with a large “hump” of effort in the middle, around the generation of production information, working drawings, schedules, etc. The general changes, such as the use of computer-aided design systems, are already operating to depress this “hump” and to shift some of the design effort towards either end of the total process (Cross, 1977).

It will be better to examine these changes at different phases of design process:

. Briefing: The briefing, sketch-design, and design-in-use stages, can take advantage of new computer models for the design and allocation of spaces and/or activities.

The computer applications do have the merit of involving the users of buildings in some of design decision making from which they are conventionally excluded. The connection between computer aids and user participation in design has been developed by Cross and Maver:

Users' involvements with their building have been in two main areas—either early in the design process, during the briefing and preliminary design stages, or very late on, actually modifying the building in use. Both of these areas need some aids if they are to progress beyond their current limitations. Principally, they need a common language for user and designer to share during the early stages, and a similar (perhaps the same) common language for user and designer to share during the continuous reconstruction of flexible buildings. The common languages could be already emerging in the predictive models of the computer programs (1973).

One of the most exciting promises of computer-aided design is the prospect of being able to sit down with a client and design a project while discussing it. In terms of complete buildings, this may be more of a client's fantasy than an architect's dream, since designers often prefer to work out ideas in private, checking for feasibility before presenting solutions to a client.

. Sketching: Sketching and drafting manually are *line-based* drawing techniques. Often it is more useful to think in terms of objects or patterns. For example, an already drawn image as a reference beneath tracing paper might be used instead of drawing it from scratch. In a computer-aided architectural design such a drawing would be *object-based* instead of line-based. Object-based drawing can allow the creation of drawing libraries, sets of parts that can be assembled into drawings. It provides rapid feedback that gives the designer new opportunities to experiment and test ideas. The ability to assemble and change drawings encourages design exploration. Seeing the implications of a change in a repeated element can remove some of the guess-work from the design process.

Drawing with computer assistance is also like multiplication, because of *mass reproduction* of drawing elements .

Many computer-based CAD systems also provide the ability to project simple two-dimensional drawings into a third dimension. Views can be selected in orthographic modes or true perspective. Images are generally displayed as *wire-frame* shapes, in which lines are used to outline planes and forms. Hidden lines behind the foremost planes can be automatically removed, providing more realistic views. Some programs allow on-screen surface shading, and some allow shadow studies by providing a user-selected light source. The appearance of three dimensional modeling for architects promises an extraordinary impact on

the design professions. It allows the creation of drawings that are *interrelated* in plan, section, and elevation, such that changes in one are immediately reflected in all the others.

. Detailing: It is important to select which parts of a drawing will be developed in detail. One of the greatest advantages of computer-aided architectural drawing is the ability to change *scale* effortlessly. Zooming in and out of a drawing can be enormously useful. Just as a designer often goes back and forth between large and small scales, drawing produced with computer assistance can be constructed in a way that might be described as cyclical. The first, simple parts of a drawing can be assembled, then returned to and developed in detail. The advantage of this cyclical process is in the creation of a drawing made up of parts over which the draftsman has a tremendous amount of control. It also permits a designer to work at several different levels of detail almost simultaneously.

Architectural drawings have traditionally been small scale representations of large objects. A computer perceives all drawings as if they were drawn full-scale, using different units. Although the CAD display screen is smaller than a typical drawing sheet, it can be used as a telescopic window into a drawing, magnifying it or shrinking it without regard for scale. In fact, the idea of *scale* is almost meaningless when working on an electronic drawing, since it can be moved from a view of a detail in second. It can be worked at *real-world* scale: it is the viewpoint that changes.

. Production Information Stage: Computerization of the production information procedures, such as computer selection and combination of standard details, computer-produced schedules and drawings, and computer-information retrieval, should drastically reduce the amount of effort needed at that stage of the process. After having enough information at hand, drawings, schedules, specifications, and cost estimates can be produced almost simultaneously. The extra effort and opportunities for error that inherent in doing these independently can be eliminated.

. The Management Pyramid: The introduction of computer systems into architects' offices would clearly bring about major changes in the method of working and in the composition of the typical office. One principal effect of computerization is a flattening of the pyramid of management hierarchy.

Whitfield suggests that,

the allocation of functions can be pictured also as the positioning of the interface or boundary between the human operators and the hardware of the system in terms of the relative amount of information processing to be performed by each part (Whitfield, 1967).

This thesis examines the peculiarities which will create this interaction and describe the steps that must be taken in the design and implementation of user-machine interface for architects.

The purposes are to present the concept and to foster the development of architect-computer symbiosis by analyzing some problems of interaction between architects and computing machines, calling attention to applicable principles of man-machine engineering, and pointing out a few questions to which research answers are needed.

In research of interface design the *creation of environments for enhanced interaction and problem solving* will be frequently alluded. Similarly, the aesthetic of an interface will be distinguished from its functionality, and the importance of the *satisfaction of an architect* will be emphasized *as a criterion for evaluation* rather than the objective analysis of the technological power of a particular system.

Interestingly, the language which will be used in the expressions comes quite directly from an examination of our physical environments and of the topics which we consider when we alter their form. They refer to the field of architectural design of buildings, a well established field in which controversies that have generated a range of different kinds of buildings in our environments as well as a history of the particular ideas.

In this thesis the topics that seem common to architectural and interface design will be outlined trying to use architectural examples and experiences as a way to make a concrete number of complex issues in the interface domain. We hope that consideration of this analogous domain might offer insights to individuals working in the design and evaluation of the interface. The intent of this thesis is to offer short-cuts to our early analyses as well as some time-saving cautions.

For this aim the second section will give a definition of user interface and locate its component in a computer assisted system. The problem faced by novice users and the required characteristics of a well-designed user interface will also be explained in this section. It will be examined from two different points of view as

the human side of the interface and the system side of the interface. The examination points of the human side of the interface will be: user interface requirements, accessibility, starting and terminating sessions, training and user aids, and the vocabulary. For the system side of the interface they will be: functionality and visual interface, system dynamics and response time, work-session interrupts, consistency and compatibility, visibility and simplicity, data organization and dialogues.

In the third section it will be looked for guidance, and the scope of the problem is to be identified. After the statement of designer's interaction tasks, the controlling tasks will be examined.

The fourth section will provide detailed information about common styles of interaction techniques -which will solve these tasks -namely command language interface, natural language interface, menu-driven interface, iconic interface, graphical interface, form-filling interface, window-oriented interface, direct manipulation, speech communication, multi-media communication and virtual reality.

The fifth section will present the effort to construct a specification method. For this reason architectural formal specifications will be examined and the reason of the need for architectural prototypes and standardization will be explained.

The sixth section will summarize the work done, evaluate the interface for computer-aided architectural design and offer a guideline which consists of the attention points both for the architect who wants to buy and use a software and for the computer programmer who wants to write a software for the architect.

2. HUMAN-COMPUTER INTERACTION

A computer system consists of three major components: hardware, software, and the user. The intersection of these components is probably the most important part of a successful system -the human-computer interface.

A user interface can be thought of as an input language for the user, an output language for the machine, and a protocol for interaction. Rissland views the interface as more than a simple “membrane”. It is not only a screen which separates the user and his computing environment. It is more than a simple “gateway” which through user input and output pass. It includes physical aids (like mouse). The interface is not only characterized by physical attributes. Rather than that, it includes aspects like the user's intentions. Schematically, this is the difference between indicating the scope of the interface as a box around both the user and machine rather than as line or a zone between them (Rissland, 1984).

According to Botterill, the term user interface is defined as “the way the software communicates or interacts with the user to help in accomplishing his/her tasks” (Botterill, 1982). This interface, then, includes the means by which the system accepts requests from the user and the way information is returned to the user. The level of ease of use depends on what user must learn and to acquire the desired end results.

Gittins et. al. (1984) define the user interface as consisting of three elements:

- . a “user model” of the system,
- . a set of “operations” that may be performed, and
- . the “media” used between the user and the operations.

The “user's model” denotes the conceptual model of the information to be manipulated and the process to be applied to the information. The degree to which the system concurs with the model is the degree to which it is viewed as user-friendly. The “operations” and the “media” are the computer component of the interface. The media types form an envelope around the operations. It serves

to effect a transfer from an internal representation of data to some external one (Gittins et. al., 1984).

With this respect, after briefly mentioning the development of human-computer interaction, this section will describe typical problems faced by new users and will define characteristics of a well designed interface.

2.1. Development of Human-Computer Interaction

Interactive computing came into widespread use in the 1960s and Human-Computer Interaction (HCI) came to have high significance for applications. By a lot of specialist researchers, it is regarded as basic concern in computer-based system design and application. Now researchers treat HCI as a distinct discipline with its own methodologies, foundations and techniques. Its focus of attention and area of change are accepted as a larger part of the total development of computer systems. Gaines and Mildred show the development in computing, artificial intelligence and human-computer interaction through seven generations in table 2.1 (Gaines and Mildred, 1986).

With respect to this table we can express that there is the introduction of new technologies at the transition between generations. In the zeroth generation electromechanical relays are replaced with vacuum tubes which can be accepted as a breakthrough in electronic device technology (EDT). In the first generation Mauchly and Von Neumann's breakthroughs brought the concept of programmability with the digital computers that are leading to the virtual machine architecture (VMA) principle. They defined computing science as a separate discipline from electronic engineering. The second generation corresponded to breakthroughs in problem-oriented languages (POLs) that made programming easier. The third generation corresponded to breakthroughs in operating systems which gives time-sharing and human-computer interaction through conversational computing. The fourth generation corresponded to breakthroughs in expert systems which allow the development of knowledge-based systems (KBSs). The fifth generation corresponded to breakthroughs in machine learning which gives inductive inference systems (IISs) and promote the current research to the learning systems. The sixth generation is still under thought. It will probably involve new technologies for high density information storage and processing which are under investigation now. It seems that the "breakthrough" into the sixth generation will come from work on robotics relating to autonomous activity systems (AASs). These systems will be goal

GENERATION	HARDWARE/ SOFTWARE	STATE OF AI	STATE OF HCI
0 1940-47	Up and Down Relays to vacuum tubes	Mind as Mechanism Logic of neural networks <i>Behavior, purpose & technology</i>	Designer as User Judge by ease of use
1 1948-55	Gee Whiz Tubes, delay lines, drums Numeric control, nav aids	Cybernetics Turing test Ashby's homeostat Grey Walter's tortoise Samuel's checkers player <i>Design for a Brain</i>	Machine Dominates Person adapts to machine <i>Use of human beings</i>
2 1956-63	Paper Pushers Transistors & core stores Control programs Fortran, Algol, Cobol <i>Communications of ACM</i>	Generality/Simplicity The Oversell Learning Machines Self organizing systems Dartmouth AI conference <i>Mechanization of thoughts Process</i>	Ergonomics Console ergonomics Job control languages Simulators, Graphics <i>Breakthrough to HCI</i>
3 1964-71	Communicators Interactive terminals Relational model	Perform by Any Means Semantic nets Scene analysis Resolution principle <i>Machine intelligence</i> <i>Artificial intelligence</i>	Man-Machine studies Interactive Experience Time-sharing services Interactive terminals Speech synthesis <i>Int. J. Man-Machine Studies</i>
4 1972-79	Personal resources Personal computers Supercomputers Very large file stores Databanks, videotex	Encoded Expertise & Over Reaction Smalltalk, frames Scripts, systematic grammars <i>Cognitive science</i>	HCI Design Rules Personal computing Dialogue rules Videotex services Altair and Apple PC's <i>Byte</i>
5 1980-87	Action Aids PC's with power & storage of mainframes plus graphics & speech processing Networks, utilities	Commercialization LISP and Prolog machines Expert system shells Knowledge bases <i>Handbook of AI</i>	User-Natural Systemic Principles Xerox Star, IBM PC Apple Macintosh Video Disk Human protocol
6 1988-93	Partners Optical logic and storage Organic processor elements AI in routine use	Learning and Emotion Parallel knowledge systems Audio and visual sensors Multi-Modeling	User-Similar Automated Design Integrated multi-modal systems Emotion detection

Table 2.1. Developments through seven generations (Gaines and Mildred, 1986)

directed and their activities will be generated by internal planning which take into account both their goals and their interaction with the environment.

The third column of table 2.1 shows the concept of HCI developments through generations. In the first generation, the operator was part of the design team. His behavior is adapted to that required by the machine. Early computers were slow, expensive and unreliable; so that interactive use was rare. Interacting with machines were a skilled operation. Operators accepted the problems of the interface as minor within all the other difficulties of using computers.

Professional ergonomic considerations of computer systems commenced in the second generation. It focused attention to the potential of the computer as a facilitator of aspects of human creativity and problem solving. First recorded paper about this concept in the literature was by Licklider (1960), who imagined a pair of human and machine capabilities that he labeled “man-computer symbiosis”. His purpose was to present the concept and to foster the development of man-computer symbiosis by analyzing some problems of interaction between men and computing machines. Licklider goes on to justify his belief that computers integrated effectively into the thought process would improve or facilitate thinking and problem solving. In a later paper, Licklider and Clark (1962) outline applications of man-computer communication to military command and control mathematics, programming, war gaming and management gaming, planning and design, education and scientific research. They report some early experiments and prototype systems that demonstrate the potential of using computers in these applications. During the same generation a number of investigators began thinking that the computer could be used to manipulate pictures as well as numbers and text; and they began exploring the potential for enhanced graphical communication between human and machine. Ivan Sutherland (1963) was successful with his work about “sketchpad” system. In developing Sketchpad, he introduced many powerful new ideas and concepts such as the concept of the internal “hierarchic” structure of a computer-represented picture, the concept of a “master” picture and of its “instances”, the concept of the constraint, the ability to display and manipulate “iconic” representations of constraints, the ability to copy as well as instance both pictures and constraints, some elegant techniques for picture construction using a light pen, the separation of the co-ordinate system and some operations such as “move” and “delete”. At the same time Coons (1963) outlined the requirements for a computer-aided design (CAD) system, Ross and Rodriguez (1963) presented the requirements for CAD in terms of languages and data structures, Stotz (1963) described the hardware requirements for CAD, and Johnson (1963) generalized sketchpad to allow input and manipulation of three-

dimensional line drawings.

The significance of HCI and its importance in time-sharing was recognized at the beginning of the third generation by the first conference on HCI; the *IBM Scientific Computing Symposium on Man-Machine Communication*, held at Yorktown Heights in May 1965. The sessions covered were, Scientific Problem-Solving, Man-Computer Interface, Languages and Communication, New Areas of Application and Man-Computer Interaction in the Laboratory. Davis (1966), Fano and Corbato (1966), and Licklider (1968) had also proposed the development of the time-sharing system as a means of allowing the computer to work on several jobs simultaneously. Sutherland et. al. (1969) suggested the tremendous potential of computer graphics which required advances in graphics hardware and software. On the software front there was progress in two major directions: Investigators at Lincoln Laboratory and other sites developed operating systems that are capable of supporting interactive graphics under time-sharing, another step towards making the technology more cost-effective. Simultaneously a number of languages were developed with embedded graphics support that facilitated the production of graphics applications. Psychologists and human factors specialists also began at that time looking more broadly at issues in human-computer interaction where they could play a useful role. Shackel (1969) and Nickerson (1969) were two representative workers for such a concept. *Ergonomics* was a special subject of the papers given at an *International Symposium on Man-Machine Systems* held in Cambridge, England, in 1969; the *IEEE Transactions on Man-Machine Systems* reprinted the same papers to remind of the same subjects and the *International Journal of Man-Machine Studies (IJMMS)* started to be published in 1969. Technical Group on Computer Systems within the Human Factors Society, was established in 1971.

While such publications provided a forum for HCI research on the variety of user experience of interactive systems applied to many tasks, the papers from commercial sources expanded the fifth generation HCI literature. By encouraging programmers to think about how they could improve their own interface to their computerized tools, and thereby increase their productivity and enhance programmability and maintainability, led them to improve user interface design. A book summarizing the first decade of this activity was that by Shneiderman (1980). The monthly publication of *IJMMS* and two new journals on human factors in computing, *Behavior and Information Technology* (1982) and *Human-Computer-Interaction* (1985) were the others. A large number of sessions of human factors meetings were devoted also to similar topics. Conference on *Easier and More Productive Use of Computing* was held at

the University of Michigan in 1981. Annual *ACM Special Interest Group on Computers and Human Interaction (SIGCHI) Conference on Human Factors in Computing Systems*, begun with the successful 1982 meeting in Gaithersburg, Maryland. *IFIP Conference* entitled Interact was held initially in 1984 and again in 1987. *British Computer Society Conference* entitled HCI began to be held annually since 1985. *Journal of Human-Computer Interaction*, began to be published in 1985.

The availability of low-cost computers with graphic displays increased their use in psychological studies. The fall in computer costs and the decreasing differences in hardware and software capabilities from different manufacturers led to increasing commercial interest for good human factors. Ease-of-use and user-friendliness began to be seen as saleable aspect of computer systems. The introduction of Xerox Star in 1981 and the Apple Macintosh in 1984 are good examples to this end. Xerox pioneered the development of congenial graphical interfaces to workstations and to applications such as text editing, creation of illustrations, document creation and electronic mail that could be supported within the workstation. These user interfaces incorporated various kinds of windows, menus, scroll bars, mouse control and selection mechanism, and views of abstract structures, all presented to the user and integrated in a consistent manner.

At the 1989 A/E/C Systems show, a major trade show of computer hardware and software for the construction industry, Autodesk, which produces the widely used CAD program AutoCAD, conducted an invitation -only demonstration of cyberspace that it described as a "virtual reality system". Special head-mounted computer displays permitted the user to enter into a computer graphics image and, by donning a special glove, manipulate objects within that computer-generated environment. At the 1990 A/E/C Systems show, other vendors introduced systems that produced similar effects. Whether or not virtual reality gains rapid market acceptance, it is time for architects to take a fresh look at how they are using computers. From the dollars and cents perspective, the low cost of personal computers has permitted architectural firms to implement computer technology over the past five years. Realizing the benefits of this investment in automation is now a business concern for most practices.

2.2. Typical Problems Faced by New Users

A user who is trying to learn a system puts forth effort in an unfamiliar environment to overcome certain types of learning difficulties. These difficulties are inevitable characteristics of human-computer interaction. They are potential problems in any system.

The novice user (as opposed to the skilled user) is the most sensitive indicator of good or bad dialogue design decisions. Observations of hundreds of causal users have shown that they are mainly concerned with knowing what kind of things they are dealing with at any given instant during the dialogue, and what can they do with them. Nievergelt and Weydert characterize the difficulties experienced by users unfamiliar with a given interactive system with the following questions:

Where am I?
What can I do here?
How did I get here?
Where can I go, and how do I get there? (Nievergelt and Weydert, 1987)

A well designed system allows the user at all times to obtain a conveniently clear answer to the above questions. In order to be easily understood, the information which the user may want to know about the state of the dialogue must be structured.

According to Carroll (1987), the type of learning environment affects how the user perceives the system and how easily she/he learns to use it. He specifies the problems that are shared by a person learning to use a system, in table 2.2.

People have difficulty to start at all, because they are *disoriented* by the screen display, by the manual, and by the bad fit of both to their own expectations. The system is unresponsive to what they do (*illusiveness*); the screen is *empty* and/or unchanging. When information does appear on the screen, it is for them like a *mystery message* and often useless. It may stay on the screen too long and confuse later work; it may flash momentarily, or be located in a remote part of the display, and be missed. Delicacies of command interpretation and command architecture make the causal connection between commands and functions appear unpredictable (*slippery*) or *paradoxical*. Invisible *side-effects* of user actions enhance this impression. Finally the system's *laissez-faire* structure allows the new user to become lost in mystery messages, commands, and side-effects.

Disorientation	The user does not know what to do in the system environment
Illusiveness	What the user wants to do is deflected towards other, perhaps undesired goals
Emptiness	The screen is effectively vacant of hints as to what to do or what went wrong
Mystery messages	The system provides feedback that is useless and/or misleading
Slipperiness	Doing the “same thing” in different situations has unexpectedly different consequences
Side effects	Taking an action has consequences that are unintended and invisible, but cause trouble later
Paradox	The system tells the learner to do something that is clearly inappropriate
Laissez-faire	The system provides no support or guidance for overall goals (e.g. “creating a program”)

Table 2.2. Typical problems faced by new users (Carroll, 1987)

Monk suggests to people with little or no expertise in computing learn how to use an interactive software package, acquire a good deal of new knowledge in order to achieve their task objectives in an efficient and effective manner. In order to invoke the operations, he proposes to learn to communicate with the system via the interface dialogue. This requires an understanding not only of the dialogue syntax but also how the domain of application is represented in the computer, in terms of systems objects, their attributes and their relationships (Monk, 1984).

The first solution approach that comes to Carroll’s mind is using “common sense”. A serious common-sense analysis of the new user’s may provide some knowledge, but it alone does not solve the problem. The more fundamental point is that people want to *do* things with computers and, particularly when they are learners, they make errors. These errors complicate the *pure* forms of the problem and are impossible to prevent (or to analyze) by mere common sense. The key point is motivation. In an exploratory environment the learner experience belongs to the learner. The environment affords, encourages, and even demands conceptual and empirical experiment. This motivational orientation overcomes the cognitive learning problems. Carroll listed the properties that should exist in the exploratory environment in table 2.3.

Responsiveness	When the user does something, he gets some feedback (at least informational)
Benchmarks	The user can tell where he is within a given episode or session. He has the means for assessing achievement and development of skill
Acceptable uncertainty	Being less than fully confident of his understanding and expertise is OK
Safe conduct	The user cannot do anything <i>too</i> wrong
Learning by doing	The user <i>does</i> so that he can learn to do: he designs a plan; he does not merely follow the recipe
Opportunity	Most of the things the user learns to do work <i>everywhere</i> . He can reason out how to do many other things
Taking charge	If progress stagnates, something new is suggested or happens spontaneously
Control	He is in control, or at least has the illusion of being in control

Table 2.3. Typical characteristics of an exploratory environment (Carroll, 1987)

These properties transform the problems of new users. Now the difference between the challenge and an obstacle can be identified. It depends on the character of the learning environment. If the learner's motivation is task oriented and if the learner feels in control of the situation, then obstacles can become challenges. A person working in an exploratory environment *expects* laissez-faire and illusiveness; regard paradox, side effects and slipperiness as interesting potential keys to the internal logic of the environment; and is calm by disorientation, emptiness, any mystery messages. Each new problem is a direct invitation to learn. In such an environment, the learning belongs to the learner.

In any case, if we assume that learners will always make some errors -no matter how good our cognitive solutions to interface design are- then the issue becomes motivating learners. This actively solves the problems they encounter.

2.3. Characteristics of a Well Designed Human-Computer Interface

With interactive computing systems there are important differences in the nature of the tasks being automated. The control and display of physical systems is being replaced by the manipulation and display of conceptual ones.

The existence of the two gulfs refer to the critical requirement for the design of the interface: to bridge the gap between goals and system. According to Norman, there are only two ways to do this: move the system closer to the user or move the user closer to the system. Moving from the system to the user means providing an interface that matches the user's needs. In that form it can be readily interpreted and manipulated. This confronts the designer with a large number of issues; because not only do users differ in their knowledge, skills and needs, but for even a single user's requirements for one stage of activity can conflict with the requirements for another (Norman, 1986).

Winfield identifies the amount of user participation in the actual design of the interface, from absolutely no involvement to total control. He illustrated this continuum in figure 2.1.

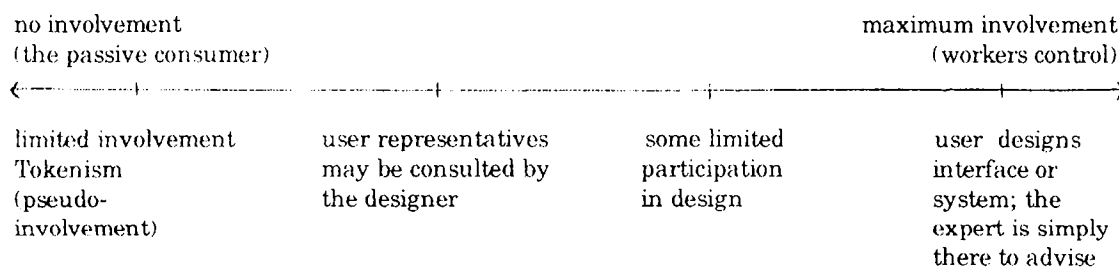


Figure 2.1. Use of human resources in interface design (Winfield, 1986)

According to him, a major force in human behavior is the desire to control. In using computers the desire for control increases with experience. Novice terminal users choose to follow the computer's instructions and to accept the computer as the controlling system in the interaction. With experience and maturity, users reject the computer's dominance and prefer to use it as a tool. The users perceive the computer as an aid in accomplishing their own job or personal objectives and reject messages that suggest the computer is in charge. So, there might be user involvement because of perceived user demands. The user can here demand the right to examine and, if felt necessary, challenge the system design plans. Effective systems generate positive feelings of success, competence, and clarity in the user community. The users are not hindered by the computer and can forecast what happens with each of their actions (Winfield, 1986).

For this aim some classes of knowledge can be inferred from the nature of the task and the system. The "primary knowledge", indicated by double borders in figure 2.2. is required for successful use of the system. An idealized user should

have these primary knowledge. As well as having information about the problem in hand, the user would have to know about physical aspects of the interface, about the interface dialogue, about the nature of the operations performed by the system and finally about the aspects of the particular problem area represented in the computer. That means the user must translate goals conceived in psychological terms to actions suitable for the system. Then, when the system responds, the user must interpret the output. He must translate the physical display of the interface back into psychological terms. With any real interaction, however, the user will call upon secondary sources of knowledge (boxes with single borders in figure 2.2.) in order to infer primary knowledge which is lacking or uncertain. During learning, these sources of secondary knowledge will have a strong influence on performance (Hammond and Barnard, 1984).

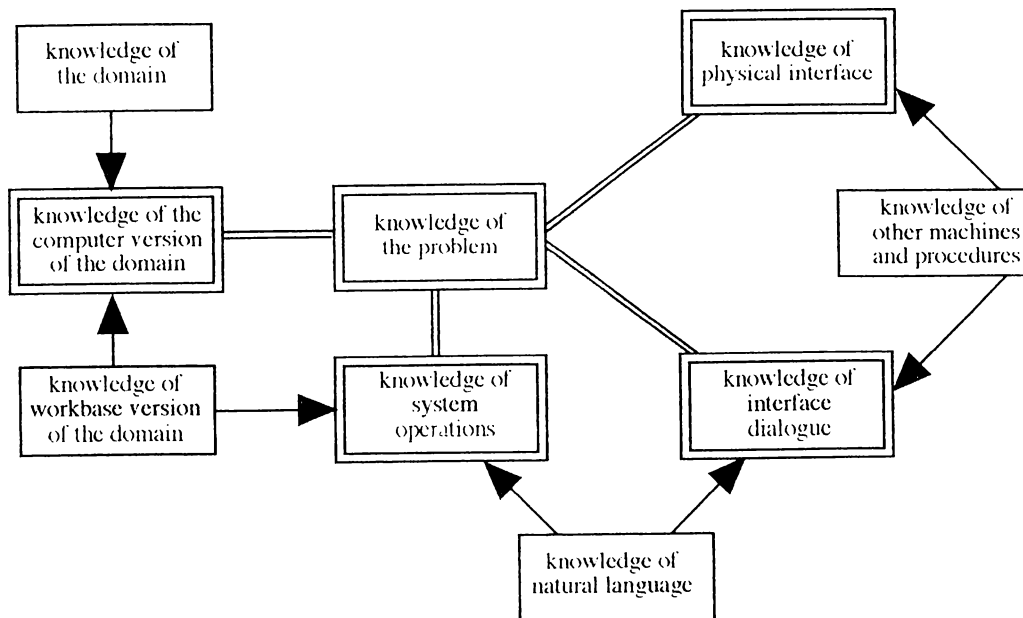


Figure 2.2. Knowledge required in an interaction. The blocks with double boundaries, connected by double lines, indicate primary information use by the ideal user; other blocks and lines indicate secondary sources of interference and facilitation (Hammond and Barnard, 1984).

Different representations allow different inferences to be drawn. Different types of knowledge can be used to deal with different aspects of the interaction. The major responsibility should rest with the system designer. He must assist to the user in understanding the system. This means providing a good, coherent design model and a consistent, relevant system image.

Norman defines three different concepts that must be considered: The conceptual model held by the designer, which he calls *Design Model*, the conceptual model formed by the user, which he calls the *User's Model*, and the

image resulting from the physical structure that he calls the *System Image* (Norman, 1986).

The “Design Model” is based on the user’s tasks, requirements, and capabilities. The conceptualization must also consider the user’s background, experience, and the powers and limitations of the user’s information processing mechanism, most especially processing resources and short-term memory limits.

The “User’s Model” is not formed from the Design Model; it results from the way the user interprets the System Image. Thus, in many ways, the primary task of the designer is to construct an appropriate System Image. He must realize that everything the user interacts with, help to form that image: the physical knobs, dials, keyboards and displays, and the documentation, including instruction manuals, help facilities, text input and output, and error messages. The designer should want the User’s Model be compatible with the Design Model. This can only happen by the interaction with the System Image. These comments place some difficulties on the designer. If one hopes for the user to understand a system, to use it properly, and to enjoy using it, then it is up to the designer to make the System Image explicit, intelligible, consistent (Norman, 1986).

Based on these considerations the following three concepts are introduced by Nievergelt and Weydert as the fundamental structuring tools for the design of man-machine dialogues:

. Site: At any moment a user wants direct access to only a small part of the data present in a system. A collection of data which interest the user for some purpose can be attached to a site. Thus it becomes a unit that can be operated as a whole in certain ways (such as copying); for other purposes data attached to a site can be regarded as being hierarchically structured into subsites. A site may be identified with the set of data attached to it and a description of its type and structure.

. Mode: At any moment the user needs only a small part of all the commands available in the system. In response to a request for a list of active commands, only these and a few general commands used for mode changing should be displayed. A larger menu only makes the user’s selection more difficult. Thus the set of all commands must be structured into a space of modes. The commands grouped together in a mode must correspond to a meaningful activity in the user’s mind. The nature relationship among these modes give the space of modes its structure.

. Trail: The order in which a user visits various sites is a relationship among the sites which is likely to be important for the current task. In order to make this relationship, which is created during a dialogue, the notion of a trail as a manipulable object is introduced. A trail is a feasible time sequence of pairs (current mode, current site), which describes a user dialogue (Nievergelt and Weydert, 1987).

In a recent review Dean notes that systems designers have been experienced to design messages so that they are: concise, grammatical, consistent and understandable. He suggests that these are the lowest common characteristics of computer-to-human communication. Messages should also be highly relevant, specific, timely and helpful. We will assume that the main forms of computer-to-human communication are to be messages or instructions presented via a VDU screen. Computer-to-human communication is likely increasingly to take the form of synthesized or recorded human speech. Many of the guidelines advocated for communication via a screen will apply to this area too (Dean, 1982).

Winfield reports these guidelines as follows:

- . System should be tolerant.
- . People should be allowed to correct errors as they make them.
- . Messages should not be over-terse.
- . Never compel people to reread.
- . Let the audience and situation dictate the message.
- . Requests for clarification or correction of input (Winfield, 1986).

Finally we can say that there are two sides of this interface: the system side and the human side. The stages execution and perception go between psychological and physical representations; and the input mechanism and output displays of the system go between psychological and physical representations. The quality of the interaction depends upon the "directness" of the relationship between these two variables. We change the interface at the system side through proper design. We change the interface at the human side through training and experience. The next sections provide detailed information about the characteristics of these two sides of the interface.

2.3.1. Human Side of the Interface

For the settlement of execution and perception stages, which have been examined in the previous section, the concept of communication must be sufficiently well motivated to understand what it should involve and why it is important. Understanding is a key function of interactive systems. It is a multidimensional quality rather than as something one has or one does not have.

Riley relates understanding to three characteristics of the user's knowledge: internal coherence, validity and integration. Coherence concerns the degree to which the user's components of knowledge are related in an integrated structure. Validity concerns the extent to which the user's components of knowledge accurately reflect the behavior of the system. Integration concerns the degree to which the components of the knowledge are related to other components of user's knowledge. The degree of internal coherence, validity, and integration does not depend on single aspect of knowledge, but upon several. This emphasizes that a user should not be considered as either performing with or without understanding. It is possible for him/her to have acquired some components of knowledge and not others (Riley, 1986).

Shackel bases a successful design for interaction between user, task and system upon five fundamental features:

- | | |
|---------------------------|--------------------------------------------------------------------------------------------------------------------|
| 1. User-centered design | - focused from the start on users and tasks |
| 2. Participative design | - with users as members of the design team |
| 3. Experimental design | - with formal user tests of usability in pilot trials, simulations and full prototype evaluations |
| 4. Iterative design | - design, test and measure, and redesign as a regular cycle until results satisfy the usability specification |
| 5. User-supportive design | - training, selection (when appropriate) manuals, quick reference cards, aid to "local experts" and "help" systems |

Table 2.4. To be successful, design for interaction between user, task and system must be based upon these five fundamental features (Shackel, 1990)

The next sections provide detailed explanations about the characteristics which will create these fundamental features of the human side of the interface.

2.3.1.1. User Interface Requirements

We know that improving particular interfaces and computer systems into different environments fundamentally alter these environments, in terms of social interactions as well as personal development. We realize that a consideration of the nature of these environments in the preparation of the systems is imperative. In addition, it is most probably the case that the cognitive processing of computer users will change this new system. Given this, Hooper asks that:

just who is the user we should be studying in designing the interfaces? Is it the current novice or current sophisticate? Or is it our guesses about future sophisticates? And do we design for a majority of people, or for an elite whom we judge to be good models of what the majority will be in the future? Moreover, how much of our effort do we put into adapting our machines to people, and how much in adapting people to our machines (e.g. in providing good tutorials)? And just how will we deal with issues of future changes? (Hooper, 1986)

In addition, Cockton's questions are as follows:

- . Is the user experienced in the use of computers?
- . Is the user experienced in the task domain of the application?
- . How regularly does/will the user use the application?
- . How long has the user been using the application? (Cockton, 1990)

According to him, if the answer to the first two questions is "no", then a supportive user interface is required. The user wants to understand the controls, all the displayed information or the significance of all the activity. The common solution to this problem is a sequence of orientation, making clear the purpose of questions, the possible answer and the significance of questions and answers (Cockton, 1990).

Users will not become experts immediately after their interaction became automatic, unconscious, skilled behavior. They will have to pass a conscious problem solving phase. Cockton's solution at this point is a good signposting in the interaction, that in some means of "suggesting" the next possible steps in an interaction, rather than "enforcing" them. The interaction may still be sequential, but modeless direct manipulation can be just as supportive for this stage of user experience. Prompting signposts are replaced by timely feedback which allows users to evaluate their problem solving and backtrack to try another plan if necessary (Cockton, 1990).

If interaction offers always sufficient cues, comments or user-requested help that reduce the user's need to progress the fully accurate and optimal skilled interaction, many users may *never* leave this problem solving phase. Often the realization of this fact is masked behind designer's and system commissioners' complaints. Most users are only using a part of a system's capabilities. So, it could be the designers commissioners who are responsible for developing a system for a typical user.

If users are already skilled in the task domain, this means that they will be coming to the system with present ways of achieving task goals. They will know where they are, where they want to be and how they can go from here to there. No technical specialists will be able to see an adequate description of a system's intended users. Another fact about expertise is that skilled performance varies from one individual to another. One expert's way of doing something may be no more acceptable to another expert. According to Cockton, the need here is "flexibility". To the requirements for sequence, flexibility, signposting, safe and profitable exploration, he adds the need for interleaved activities. He also emphasizes that the level of interleaving is important (Cockton, 1990).

Finally, the most important element is the *user*. The golden rule is: "use the user". Test the message out; if need run a controlled experiment. Do not assume that the user is a passive static system to be controlled, modeled and directed by the computer. Evaluate all actions of the system in terms of their effect. Counsel the users, listen to their comments. Redesign the communication in the light of these (Winfield, 1986).

2.3.1.2. Accessibility

The users would like to have immediate access to the computer on a continuing basis. They would like to be able to work on a problem whenever, and for however long as, they wish. This is a difficult objective for many systems, but to provide maximal access should be an objective of the designer. According to Fallon there are four elements which will create this aim (Fallon, 1990):

First is the "availability" of computer workstation. You cannot maintain all information in computer environment unless everyone has access to a workstation whenever needed. This means a 1:1 ratio between people and machines.

Second is “training and support”. Everyone must receive sufficient training to be completely comfortable working within the automated environment.

Third is the issue of “ease of use”. The interface should reduce the level of programming expertise required to program applications and manage the system. In this way the system functions would be easier for them to use. The interface should also increase ease of use for the end user who is not a data processing professional. It must give the programmers what they need to conveniently produce easy to use applications for the end users. This should provide the end user a simple way to request applications, enter data, and request the results (Botterill, 1982).

Fourth is the question of “reliability”. Working in a computer based environment, hardware, software and network failures will prove equally disastrous. Reliability should be a major selection criterion for all three components of the computer infrastructure.

2.3.1.3. Starting and Terminating Sessions

The user requires some time to access the particular software system. During that time he wants to interact and to initialize the system for the needs of his/her particular work session. This time should not involve consuming preliminaries.

A lot of users may use the same tools on different occasions. According to Nickerson the user identification should be sufficient for initializing the session. The system should have the capability to bring the user to the point he has left and automatically re-establish the software when the user returns to the computer after the termination of a work session. At termination, the user should not have disconnect every connection that was established in starting the session (Nickerson, 1981).

2.3.1.4. Training and User Aids

There are many systems in existence that are easy to use and that provide the users to have the necessary experience with, and understanding of, them. Typically, however, such knowledge and experience are gained only at a considerable cost in time and effort. The problem is to provide to potential users

the information they need to determine how much they must learn and provide them when and if they require it.

New users beginning work with a system require assistance to get started. According to Baecker and Buxton this can be done through “teaching” them the relevant principles of operation, and/or through “training” them in the skills required for successfully carrying out the desired tasks. New users may be *novice* or *expert* in a particular technology, and the approaches required for teaching them will need to be dramatically different. Users also bring different kinds of expertise to bear upon a situation. Depending upon their motivation and upon their readiness to certain kinds of tasks, users may also be receptive or closed to absorbing instruction (Baecker and Buxton, 1987).

Nickerson identifies two types of training material that are desirable for any person-computer system: one which will introduce one to the system and another which will facilitate the advancement of a user from novice to expert status. It is not reasonable to expect that a novice user should be able to exercise the full power of a system the first time he uses it; but he should be able to do something he perceives as nontrivial and helpful. Introductory training material should be designed, and bring the beginner to the point of interest quickly. The further training that is necessary to increase the user’s capability with the system should be provided both by conventional documentation and by training facilities incorporated within the system itself. Ideally, the training facilities incorporated within the system should include the ability to monitor a user’s skill level and volunteer information in order to encourage and facilitate the acquisition of new knowledge and expertise (Nickerson, 1981).

Brown explains his thought on this problem in the following quotation:

Perhaps more important, we need to consider designing instruction explicitly to help users develop strategies for ongoing learning -strategies for drawing on themselves and others as resources such as documentation in ways that help them extend their understanding. In order to help users become comfortable with guessing as a mode of extending their knowledge, we might purposely teach incomplete submodels of a system and then provide student with problems that force them to use those models as a basis for deriving solution strategies (Brown, 1986).

Schneider (1985) proposes a new set of working guidelines to promote the training of high performance skills:

- . Present information in a consistent manner.
- . Allow numerous trials of critical skills.

- . Do not overload short-term memory; do minimize memory decay.
- . Vary aspects of the task that vary in the operational setting.
- . Maintain active participation of the trainee.
- . Maintain high motivation of the trainee.
- . Present information in a context that illustrates in a several points at once.
- . Intermix training of various component skills.
- . Train under slightly speed-up conditions.
- . Train strategies that minimize the workload of the operator.
- . Train skills for time-sharing under situations of high workload.

2.3.1.5. Vocabulary

There are some strategy differences between the vocabularies used in constructing the interfaces. These differences are not only a function of the type of command set. The fundamental point here is that different types of users have different cognitive strategies. These distinctions influence their individual dialogues with the interactive system. The different types of command names had an effect over and above these individual differences in cognitive strategy. The content of the vocabulary must modulate the individual user's predispositions for controlling the exchange of information between the system and the user.

Furnas et. al. state that people use a surprisingly great variety of words to refer to the same thing. The study of spontaneous word choice for objects in five application-related domains showed that people choose the same term with a probability of less than 0.20. The popular approach in which access is via one designer's favorite single word will result in 80-90 percent failure rates. The data obtained in the experiments shows that there is no one good access term for most objects. It follows that, there can exist no rules, guidelines, or procedures for choosing a good name in the sense of "accessible to the unfamiliar user" (Furnas et. al., 1987).

Regardless of the number of commands or objects in a system and whatever the choice of their *official* names, the designer must make available several alternate verbal access routes to each.

The inherent ambiguity of user's own words, may cause the actions not compatible with the users intentions. To avoid this, either the user can be made to memorize precise system meanings, or the system and the user may interact

to identify the precise reference. When users invoke it with their own words, the system would pick its best guesses, and present them in a menu. Each of the guesses would be labeled by some *standard* access terminology and be accompanied by a description of the standard referent. Actual execution would always be via the standard *name*, thereby to avoid the disambiguation, the learning of precise terms are required.

Kenzie summarizes the main features of the vocabulary that should be considered in constructing the messages as follows:

- words: special categories are:
 - prompts (to signal that the user or computer is ready to send or receive a message)
 - actions
 - objects
 - conjunctions
 - punctuations
 - terminators (to signal the end of a message or part of a message)
- abbreviations (shorthand words)
- characters (the sub-units of words) (Kenzie, 1988).

Wright defines that, computer-based writing tools can help to detect some difficult terminology. For example, long words are more unfamiliar than short ones. Nouns created from existing verbs (for example reduction from reduce) are usually longer than the verb forms and make the comprehension more difficult for readers. Words involving negation either explicitly (not, un-, dis-) or implicitly (decrease, reduce) can cause readers more difficulties than their antonyms. As structure of the dialogue, he suggests sequences of steps be mentioned in the order in which these steps will be carried out by the user. "Do this than do that" is a safer communication than the equivalent "Do that after doing this" (Wright, 1988).

2.3.2. System Side of the Interface

The failure to provide information and get the job done, not knowing what options are available or what is happening, frustrate the user. In the ideal case, not much more effort is required. The system must allow the user to obtain a convenient way to realize his/her task. The problem is to design the system by means of compiling the criteria according to the user's intentions. The next sections give detailed information about these criteria.

2.3.2.1. Functionality and Visual Interface

A primary consideration in the design of an interface for a computer system is that it *works*, that it fulfills the purposes for which it was intended. If a system is developed to meet the needs that exist only in the mind of the developer, the target users may fail to use it. The potential users must be involved in the initial design, so that the system developer must know what the users will need and deal with the functionality of the computer application.

According to Kammersgaard, dealing with the functionality of a computer application means dealing with what can be done with a computer application, with the set of possible products and with the purposes for which the computer application is valuable for the user. An application is developed to fulfill some purposes within a domain. Functionality relates to tasks performed within the domain. It is primarily characterized in relation to these tasks (Kammersgaard, 1990).

Nelson defines these tasks as follows:

The frequent tasks are easy to determine, but the occasional tasks, the exceptional tasks for emergency conditions, and the repair tasks to cope with errors in use of the system are more difficult to discover (Nelson, 1987).

Task analysis is central, because systems with inadequate functionality frustrate the user and are often rejected. If the functionality is inadequate, it does not matter how well the human interface is designed. Nelson emphasizes that excessive functionality is also a danger. Probably the more common mistake of designers is to make the implementation, maintenance, learning, and usage more difficult by clutter and complexity (Nelson, 1987).

Hooper argues that the designers only have had access to computing resources, and they have not considered the relationship between the functionality and visual interface. For example windowing systems provide a deviation from other systems, as do systems that focus on graphical as opposed to textual interactions. Therefore we can say that different forms may represent the same functionality. In this way we need to provide the relevant information for the task at hand, and to provide the most articulate, and appropriate manner possible. In doing it we exploit the human capacity for perceiving structure and organization, in short for understanding. We must consider carefully how graphical and textual elements relate on this visual channel, that means on the two dimensional screen. An interface will not be effective unless the

functionality of a system is revealed directly; because the interface infers the structure of the computer system and informs the user about the particular system (Hooper, 1986).

Szekeley insists on the minimization of the dependencies between the implementation of an application's functionality and the visual interface. He believes that the ability to change an application's visual interface without impacting the implementation of the functionality is crucial; and explains the benefits of separating the functionality and visual interface as:

Multiple visual interfaces can be developed for a single application, each one tailored to a different class of users, or to a different set of input and output devices; the functionality of an application can be called from another program directly, without simulating the input required by the visual interface; the visual interface can be specified by means other than programming, for example, by interactively drawing and demonstrating how the interface should behave (Szekeley, 1987).

Other works of the researchers about visual interfaces can be found in following references:

- . Bowman provides a systematic introduction to the visual language. He describes with an extensive design library of examples of showing *what*, *showing how*, *showing how much*, and *showing where* (Bowman, 1968).
- . Dondis stresses that the process of *composition* is the most crucial step in visual problem solving. She views the compositional technique as most important contrast (Dondis, 1973).
- . Marcus demonstrates the importance of careful selection and arrangement in using typography, signs and symbols, charts and diagrams, color, and spatial and temporal arrangement (Marcus, 1983).

2.3.2.2. System Dynamics and Response Time

One of the chief determinants of user satisfaction with interactive computer systems is response time. A second determinant is the variability in response time. Task characteristics and individual user characteristics interact with response time. Many computer professionals believe in the simple principle that faster is always better. Evidence from several IBM studies and other sources suggests that programmers are more productive when system response time is

kept within the one second range or even faster. On the other hand, isolated studies have shown that in some computer-assisted instruction, complex order entry, and introductory session with novices, rapid performance leads to poorer learning, less effective decisions, higher error rates, and occasionally decreased satisfaction (Nelson, 1987).

Winfield also adds that response time is the “thinking time” of the terminal user. For complex decision making there is some evidence that locking the terminal for a short period, may improve user performance on the decision making and increase user satisfaction. If users perceive the computer as a tool they may be more willing to take their time and reflect on decisions. If users feel they are involved in a dialogue in which they must respond promptly, anxiety and poorer performance may result (Winfield, 1986).

As second determinant of user satisfaction with interactive computer systems Winfield suggests to minimize the variability in response time. It is well known to all users that increasing the variability of response time generates poorer performance and lower user satisfaction. When there is a report on a programs reaction to input, the response can be message of one of three types: that the input is being processed and results are forthcoming, that there is a delay, and that the computer is unable to deal with it. People have a need for “closure” in tasks: i.e. the feeling that portions are completed. If there is a delay people need to be told whether the information has been accepted or not. Moreover if the delay continues beyond that expected they need to be told that processing is still under way and the system is not malfunctioning. If the system cannot accept the user's input the user needs to know it immediately (Winfield, 1986).

2.3.2.3. Work-Session Interrupts

System crashes are important problems for both system designers and researchers interested in the study of human-computer interaction. The designer's solution must be not only minimizing the frequency and duration of crashes, but also making the effect of crashes as harmless as possible. According to Nickerson, if it can be warned that a crash will occur, it will be very helpful. Minimizing the negative impact of a crash, in terms of lost work, by backing up files is also helpful. Providing the user with some indication of how long the system will be non-operational, may relieve the annoyance (Nickerson, 1981).

2.3.2.4. Consistency and Compatibility in Interaction

The initial working of an interface can be formulated in terms of the notions of consistency and compatibility. These terms are potentially confusable, so the distinction requires clarification. Barnard et. al. make it as follows:

Consistency refers to relationships within the components of a user's representation of operators and operations were to agree in some way, then this agreement termed as consistency. Compatibility refers to relationships between e.g. natural language representations of the operators and operations. However, unsystematic labeling of operations should be described as inconsistent rather than incompatible (Barnard et. al., 1981).

Baecker and Buxton define consistency as the use of existing skills that they call "skill transfer". That is, in a well designed system, when user is confronted with a new situation, all of the feedback mechanisms will say "this is like this other task which you have done before", and the user will be able to transfer what has already been learned, to the new situation (Baecker and Buxton, 1987).

In general, consistency and compatibility are known as the basic and useful ergonomic principles. A system which provides the user with a consistent representation within a particular type of knowledge is easier to learn and less prone to error than an inconsistent one. Since it allows the user to follow the same procedures, invoke the same command and display code, interact with the same format on the screen or window and interact input devices in the same way in spreadsheets and drawings. The problem of lack of consistency which occurs when large software systems are collections of components developed by different designers constrains users to think in terms of systems and subsystems.

Similarly, incompatibilities between system characteristics and types of knowledge of the user cause loss of time and accuracy. So the format of displayed information should be clearly linked to the format of the data entry. Another issue with compatibility is other computer and noncomputer systems that the user may be using. Small differences among systems can cause annoyance and dangerous errors. Gross differences among systems require substantial retraining and force the users. Incompatible storage formats, hardware and software versions cause frustration and delay. So Nelson concludes that "designers must decide whether the improvements they offer are enough to offset the disruption to the users" (Nelson, 1987).

2.3.2.5. Visibility and Simplicity

Boulay et. al. describe simplicity and visibility as two important characteristics for novices. According to them, novices start programming with very little idea of the properties of the system which they are going to learn to use. To help them learn these properties, the system should be simple. It should consist of a small number of parts presented in a way that can be easily understood. This can be done by analogy to other mechanism with which the novice is more familiar. Visibility is concerned with methods for viewing selected parts and processes of the system in action. Expressing the characterization of the system in either text or pictures on the user's terminal provides visibility (Boulay et. al., 1981).

Wright's point of view is that the physical appearance of visual information can affect both the legibility and the interpretation of the material displayed. Space can give a visual grouping of functionality of related elements. Appropriate use of space specifies the legibility of information. Design features are not good or bad in themselves; much depends on the way they are used. The position on the screen may lead readers to suppose that the information is distinctive; for example text under an illustration may be thought as a caption and perhaps ignored for that reason (Wright, 1988).

2.3.2.6. Data organization

Mantei and Haskell illustrate that more than half of the problems encountered by an individual in learning a system had to do with the data organization, and particularly with its perceived incompleteness and its lack of "user orientation" (Mantei and Haskell, 1983).

The transfer of data among computer applications should be automated. This takes the elimination of redundancy one step further. In fact, the emerging trend is toward dynamic data exchange, e.g. the automated extraction of quantities and updating cost estimates whenever CAD drawings are changed.

Data should be reduced and validated. One very usual mistake is failing to distinguish between data and information. Fallon provided a succinct and useful distinction: "Information is data endowed with relevance and purpose" (Fallon, 1990).

Some problem arises because of the various expertise of users and different needs of various tasks. As a result of that there must be a large class of useful data types. Beginning users need to be able to read something that introduce concepts and information in a logical sequence and that the answers of the questions that they do not know to ask. For this aim there must be tutorial guides that will help them. More experienced users need an information source where they can find answers of specific questions. In that case there must be principles of operation manuals, reference manuals and command summarizes for them. The organization, form and content of these kinds of data should improve the user interface of many systems. They should be clear, accurate, complete, well organized and current. When users need assistance during the work, what is needed could be provided on-line.

2.3.2.7. Dialogues

For many dialogues, Hammond and Barnard characterize the exchange of information in terms of its “style”, “structure” and “content”. The term style refers to differences in the character and control of the information exchange. The use of command languages, menu selection, question answering, query-by-example, and spatial control, represent different styles of dialogue. The term structure refers to the formal description of dialogue elements, their ordering within and between dialogue exchanges and the concept of dialogue content describes the semantics of the information exchange. They suggest to establish guidelines concerning the style, structure and content of human-computer dialogues for different applications and user population. They formulate the questions that must be thought as follows:

e.g. is it more appropriate to use a menu-oriented system or a command-oriented system for this application or that user population? (style); how should we order the elements of dialogue? (structure); or, what kind of command vocabulary is appropriate? (content) (Hammond and Barnard, 1984).

Maguire (1982) summarizes other recommendations as shown in table 2.5.

Area of consideration	Recommendation
Handling different levels of user	. Novice users will need explanatory dialogues while more experienced users will require a briefer form of interaction. Systems should thus contain two levels of dialogue.
Input precision	. Make use of input data expressed in vague terms.
Non-verbal signs	. The dialogue may be enriched by the use of bells, bleeps, reverse video and flashing characters. . Clear layout and the use of lower case lettering will improve the appearance of displayed data.

Table 2.5. Dialogue design recommendations (Maguire, 1982)

3. GRAPHICAL INTERACTION AND CONTROLLING TASKS

The aim of interaction systems is to provide a responsive and favorable user-computer communication. These interaction sequences can be decomposed into a series of basic interaction tasks. Therefore, the most important elements in the design of user-computer interfaces are the identification of interaction tasks, the statement of controlling tasks, and the selection of interaction techniques and devices which will perform these tasks. The purpose of this section is to offer a systematic structure which will aid the designer in the identification of the interaction and controlling tasks.

“Interaction task” is defined by Foley et. al. as “an entry of each symbol by the user, performed by means of an interaction technique” (1990). Each task has certain requirements associated by the context of the application and the characteristics of the user. In addition, the same task can be implemented by many different techniques. The function of the system’s designer must be the selection of the interaction techniques that best match both the user’s characteristics and the specific requirements of the tasks, and the selection of appropriate devices for these tasks. In some cases the devices are predetermined by the hardware procurers, not by the user interface designers. Those cases limit the set of interaction techniques that can be considered by the designer. On the contrary, when device selection is part of the design process, the designer can make a link between the techniques and the hardware prerequisites.

In making the best decomposition of interaction sequences with some basic interaction tasks, the system designer must take into account some types of human processes. Foley et. al. (1990) state these processes as perception, cognition and motor activity.

Most interaction techniques start with visual perception. The way of displaying the information can quickly locate the items that user needs. The methods such as color coding, spatial coding, blinking, brightening are important elements for the specific parts of the display. Issue of display brightness, flicker, line thickness and character font and sizes are also relevant.

The study of cognitive process shows some advantages to structure hierarchical menus to present the number of choices to use, the types of words to name and abbreviate commands. If the information concerns some categories or concepts that the user already understands, he can learn rapidly; if not he learns slowly. Using symbols or names already known, grouping the choices in several logically related subsets will bring a legible communication.

The motor process plays an important role in the reception and decision of how to respond to the stimuli and in the responses to physical actions. This may involve the movement to a particular point on the screen with picking as a stylus, moving it to the tablet. In general, the design goal must be to minimize the time taken by each of these processes. In addition, identification of these processes is an important concept.

For the performance of a complete action, some series of tasks must be carried out as a single unit. Sequential actions should be grouped into action concepts. For example a user should be able to select an object, position it at the desired location and attach appropriate labels to it.

For this reason this section will provide some conductors for the actions. Firstly the guidances will be proposed, secondly the scope of the problem will be identified, thirdly the measures of ergonomic qualities will be examined. After these, the interaction and controlling tasks will be examined.

3.1. Looking For Guidance

The various kinds of interaction techniques which have different purposes have implemented with some device such as a tablet, joystick, keyboard, light pen, trackball, etc. The properties of these techniques, their advantages and disadvantages according to some specific tasks will be examined in the fourth section. Before that, we must look at some guidelines. Foley et. al. offer three basic sources of information:

1. Experience-based guidelines.
2. Experiments with interaction techniques.
3. The human factors literature, especially that dealing with equipments design (Foley et. al. , 1990).

There are some papers explaining various interaction devices and techniques that represent a source of guidance. Some special interest group for Human Computer Interaction (HCI) such as Special Interest Group on Computers and

Human Interaction (SIGCHI) and some conferences such as those organized by Human Factors Society have also began to serve as focal points. In addition, the growing human factors literature is a promising source of guidance. These guidelines have been searched in detail in the second section under the title of the development of human-computer interaction. The common objective of all these works is to achieve functional effectiveness of both the physical equipment and the facilities that people use.

Human factors research has always confronted some methodological differences. Both the collection of data from people who actually use computers and the conduct of human experimentation that can be applied to the design of man-machine systems posed problems.

In addition, these different sources of guidance are not usually in the same disciplinary jargon and they use different terminology. Our aim must be to integrate a significant and useful body of the experiential and experimental conclusions in a unified and logical structure inferred from all these sources.

3.2. Scope of the Problem

The designer of an interactive system must define everything about the user-computer interface. He must specify various factors from the concepts that user must understand, to the finer details of screen formats, interaction techniques and device characteristics.

Firstly, he must understand the application area and user's type. The currently treated applications can aid to him/her. Hornbuckle (1967) comments on this by saying, "observing what man does normally during his creative efforts can provide a starting-point for the designer". Hansen's (1971) advice is "Know the user - watch him, interact with him, learn to understand how he thinks and way he does what he does". This process is defined as "requirements definition" or "task analysis" which has been examined in detail in the second section.

The aim of this process has to be defining the capabilities of the system that can be best presented to the user. The analysis must identify the type of the user for whom the system will be designed. This also will identify the language of interaction between computer and user. Foley et. al. define the input language as "starting with the user's conceptual model, then the command structure, the syntax and finally the assignment of physical devices and activities" (Foley et. al., 1990).

The conceptual model of the user concerns the detailed semantics of the components. Semantics is the meanings of the modifiers of the language. The syntax is the way of the assemblage of the units. The lexical design is the selection of hardware devices and of the interaction techniques by which the devices will be used. The distinction between the syntactic and lexical designs is that the syntactic design is device independent, while the lexical design is device dependent (Foley et. al., 1990).

The aim of this section is not to make the physical design of the interaction devices, such as key shape, light pen diameter, etc. They are in the scope of the human factors' researches. We will consider these devices as their characteristics under user computer communication, not as their hardware characteristics.

3.3. Measures of Ergonomic Quality

In an effective interaction design a user must be able to do his work with minimum mindful attention, and maximum effectiveness. The ideal design must minimize some psychological blocks such as boredom, panic, frustration, confusion and discomfort.

The context of the tasks and the techniques by which these tasks will be implemented are significant. The technique selected may provide simple use of physical input devices, modify the device characteristics, make the process more natural, more interactive, easier and more satisfying. This section examines the criteria which will determine the quality of the interaction techniques.

3.3.1. Primary Criteria

The primary criteria which will identify the quality of an interaction design is proposed by Foley et. al. with three important items: the time, the accuracy and the pleasure. The time is the duration that user spend to realize a particular project with the intended system. The accuracy is the notion with which the user can accomplish the project. The pleasure is the item which maximizes the creativity (Foley et. al., 1990).

The characteristics of the physical devices influence the relationship between a task and its appropriate technique. In the same technique, but in different

degrees, the primary performance criteria depend on some several factors. Foley et. al. summarizes them as follows:

- . the content of the task, and the existence of some task sequencing patterns.
- . the experience and knowledge of the user.
- . the physical characteristics of the device (1990).

3.3.2. Secondary Criteria

The primary criteria examined in the previous section must be influenced by a number of secondary criteria. Foley et. al. give a list of them as follows:

- . learning time
- . recall time
- . short-term memory load
- . long-term memory load
- . error susceptibility
- . fatigue susceptibility
- . naturalness
- . boundedness (1990)

. Learning and Recall Time: Users spend some time to learn the properties and the abilities of the technique. The time that users spend to learn the patterns that used to identify the elementary figures and sounds of a particular technique is “the perceptual learning time”. The time that users spend to learn to use the technique to achieve the desired effect is “the cognitive learning time”. The time that passes in achieving the necessary physical skill to carry out the action is “the motor learning time”. The time that measures the ease with which a user regains competence after a period of disuse of the technique is “the recall time”. These time can be measured as the skill level of the users that allow them to apply the technique in a practical sense. The allegation of these time will affect the task time.

. Memory Load: User’s memory has two forms of load: short-term and long-term memory load. If the user is obliged to remember the unprompted knowledge of some task elements it can be said that the technique has a high short-term memory load. If the technique conserves to have the sense of touch using physical devices such as mouse, it augments the load on short-term memory; because these devices are out of the field of user’s view and his/her hand has to be able to hang on them with minimum effort.

Recalling the details to use the technique requires long-term memory. This occurs in learning the key symbols of a technique, such as menu list; and in remembering the shape and identity of objects to be manipulated during some sequences of tasks. Decreasing number of steps and amount of key information, applying regular patterns to call techniques, prompting the actions and the data reduce long-term memory load.

The amount of memory load influence the learning time and skill of the user. If short-term memory load exceeds the capacity of the user, poor performance and frustration can occur. If long-term memory load augments, learning and recall time will be long.

. **Fatigue and Error:** A lot of causes may result in the sense of fatigue. Foley et. al. summarizes some of these causes as follows: insufficient variety in a regular task, uncertainty and unrealistic memory loads, poor mechanical design and uncomfortable position of physical devices, etc. (Foley et. al., 1990)

The end results of these facts such as low attention and slow reflexes affect user's satisfaction, pleasurability and task time harmfully.

. **Convenience:** The criteria of naturalness and boundedness can be grouped under the heading convenience. The transfer of activity from daily exercises may be defined as naturalness. For perception, naturalness refers to the visual forms. For cognition, it refers to the appropriate order of the facts and data. For motor activity, it refers to the devices with surroundings and context.

The size of the space where user must work can be defined as boundedness - perceptual, cognitively and mechanically. A physical limited space where users' eyes try to reach to the relevant information and ears to adjust the sounds, is perceptual boundedness. The intellectual space such as ideas, concepts, facts is limited by cognitive boundedness. The distance that user's limbs must move to use the technique is defined by mechanical boundedness (Foley et. al., 1990).

3.4. Interaction Tasks

Although there are lots of primitive action units performed by the user, the six fundamentals of them -namely select, position, orient, path, quantify, text- that are used especially by architects will be examined.

3.4.1. Select

The alternatives with which a user can make a selection are the following:

1. Menu selection with a light pen.
2. Menu selection with a cursor controlled by a tablet or mouse.
3. Type-in of name, abbreviations or number on an alphanumeric keyboard.
4. Programmed function keyboard.
5. Voice input.

Right along with commands, the information presentation of the application formed with displayed entities may also be a set of alternatives. These entities may be the symbols that represent equipment or positions. Similar interaction techniques to those for command selection can be used also for this case. (Figure 3.1.) They are as follows:

1. Pointing with a light pen.
2. Using a cursor controlled by a tablet or mouse.
3. Type-in of the entity name.
4. Pointing on a touch-sensitive panel.
5. Voice input of the entity name.

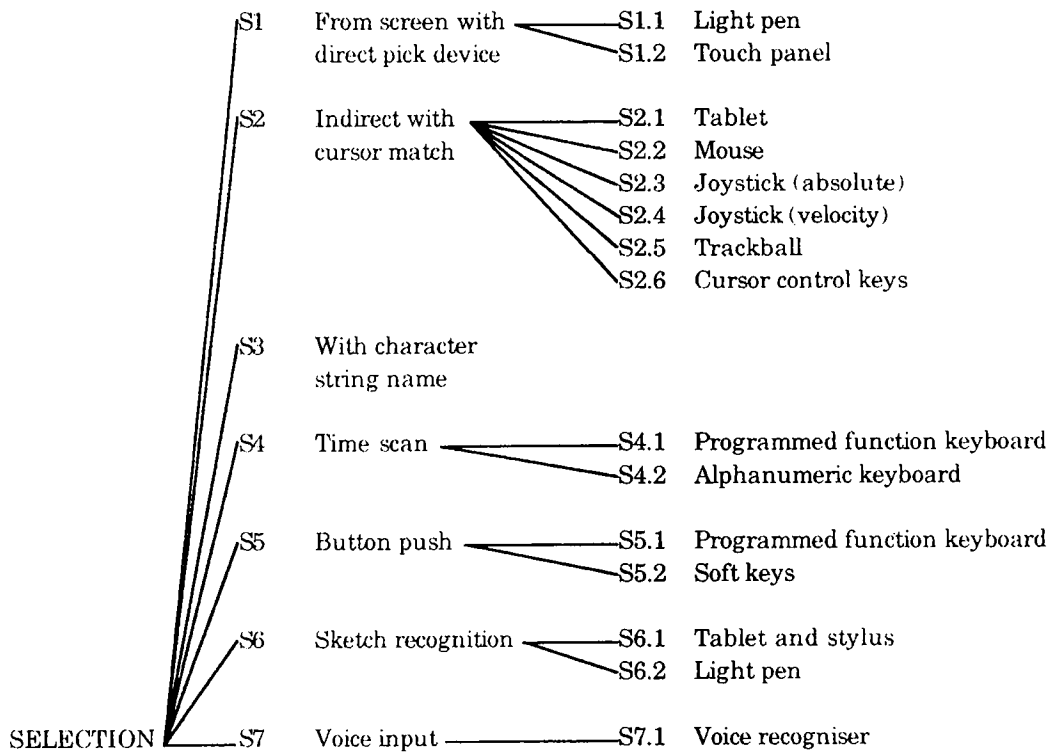


Figure 3.1. Selection techniques (Foley et. al., 1990)

The application requirements for a selection task are:

1. Size of the set from which the selection is made, if size is fixed.
2. Range of set size, if variable.

A fixed set with two choices (such as “yes” and “no”) or a large set with variable-size of displayed entities may be different techniques for selection.

3.4.2. Position

To place an entity at its particular position, the user has to be carrying out a positioning task. Commonly used interaction techniques for positioning are (Figure 3.2.):

1. Use of a cursor controlled by a tablet, mouse or joystick.
2. Type-in of the numeric coordinates of the position.
3. Use of light pen and tracking cross.

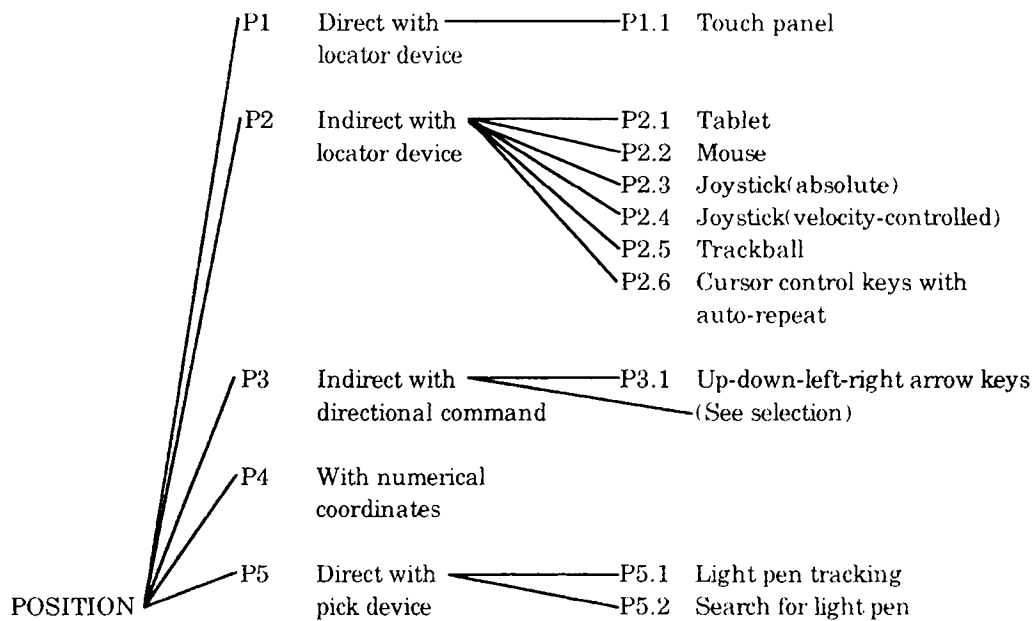


Figure 3.2. Positioning techniques (Foley et. al., 1990)

The application requirements for a positioning task are:

1. Dimensionality: 1D, 2D or 3D.
2. One loop or closed loop: If the user knows in advance the exact coordinates of the position, the visual feedback is not a fundamental part of the process; but

if he adjusts the position to obtain the desired visual result, the visual feedback is important.

3. Resolution: The accuracy over the maximum range of coordinate value specify the resolution.

3.4.3. Orient

An entity may be oriented in 2D or 3D space. In 2D, rotation can be used for orientation. In 3D, the control of the pitch, roll, and yaw of the view are necessary. Useful interaction techniques for orientation are (Figure 3.3):

1. Control of orientation angle(s) (one angle for 2D, up to three angles for 3D) using dial(s) or joystick.
2. Type-in of angle(s) using alphanumeric keyboard.

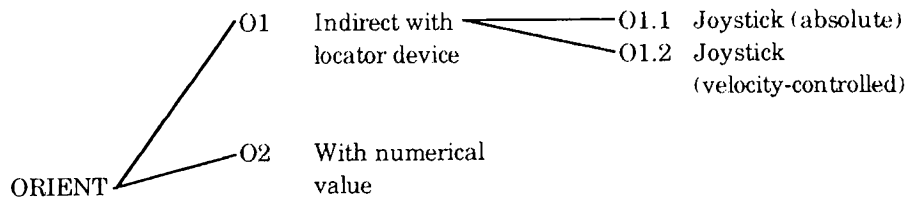


Figure 3.3. Orienting techniques (Foley et. al., 1990)

The application requirements for a orientation task are:

1. Degrees of freedom: In 2D space, a single degree of rotational freedom; in 3D space two or three degrees of freedom are available. In 3D, one degree of freedom makes a rotation about an arbitrary axis.

3.4.4. Path

Generating a path is a series of positions or orientations. Because it consists of other primitive tasks (position or orient) it is a fundamental task. Another basic dimension is “time” which changes the user’s perception. While the position and orientation task attract the user’s attention on a single action, path generation which is a series of positions or orientations and their order focuses the attention on multiple actions.

A user can generate a path of position by digitizing a sketch, indicating a route on a circuit board or showing a route on a map. He can generate a path of orientation by a similar process over the model.

The interaction techniques for generating a path are:

1. Positioning task techniques that involve use of a tablet, mouse, joystick and/or dials.
2. Orientation task techniques that involve use of a tablet, mouse, joystick and/or dials.

The application requirements for a path task are:

1. Position or orientation task along the path.
2. The interval between each element on the path defined by the time or distance.
3. Dimensionality: 2D or 3D.
4. Open loop or closed loop.
5. Resolution.
6. Type: position, orientation or both.

3.4.5. Quantify

The quantifying task is a measure, i.e. the height of an entity, specified by a value. Typical interaction techniques for a quantifying task are (Figure 3.4.):

1. Value type-in on a keyboard.
2. Rotary or slide potentiometer.

The application requirements for a quantifying task are:

1. Resolution: the number of resolvable units expresses the resolution.
2. Open loop or closed loop.

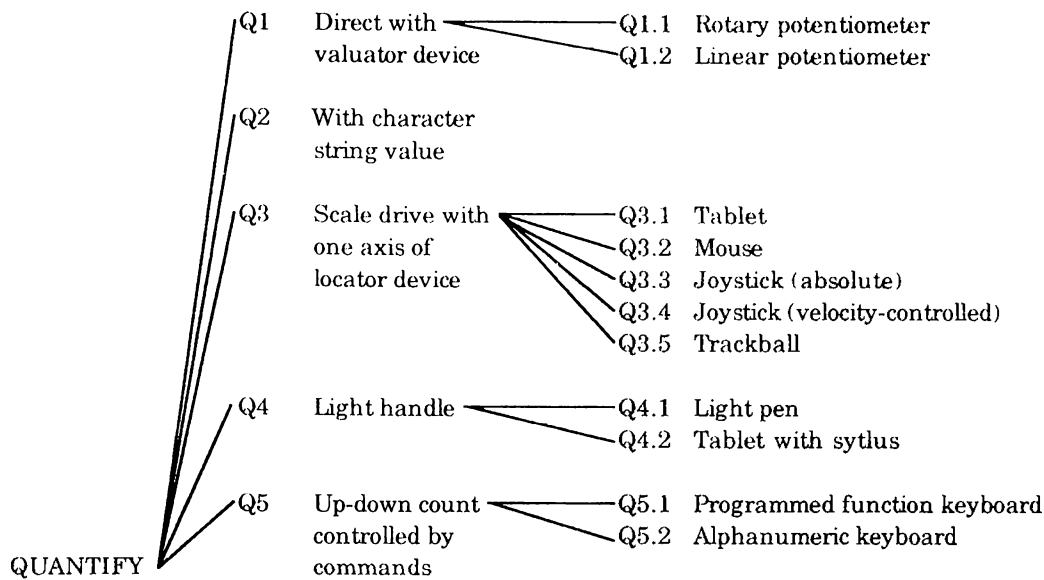


Figure 3.4. Quantifying techniques (Foley et. al., 1990)

3.4.6. Text

The text input is another interaction task. The user inputs a text string as a commentary for a drawing or as a part of a page of text. It is stored in the computer as an information. It does not serve to a command, position or orientation.

Typical interaction techniques for text input are (Figure 3.5.):

1. Type-in from an alphanumeric keyboard.
2. Character selection from a menu.

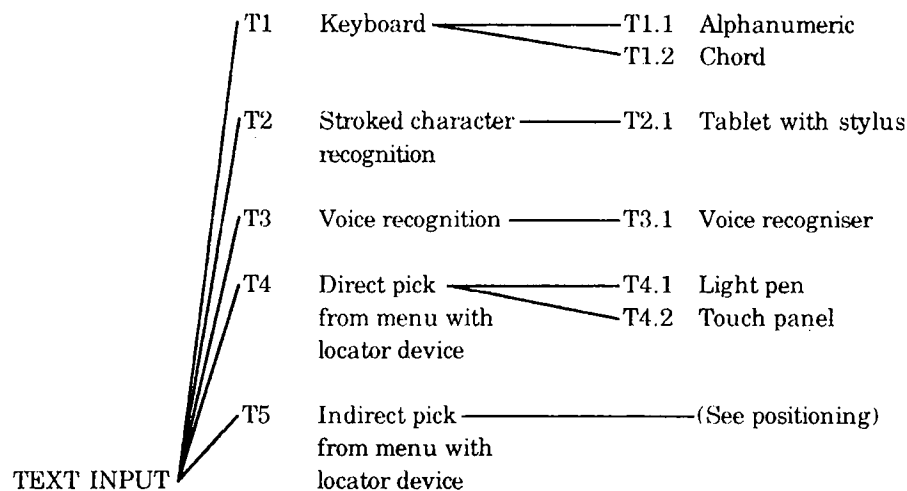


Figure 3.5. Text-entry techniques (Foley et. al., 1990)

Another requirements for text input task may be a specific character set. However it does not affect the choice of technique or device.

3.5. Controlling Tasks

Fundamentally, the concepts of each interaction task such as select, position, orient, path, quantify, text is choosing. An entity among a set can be chosen by the selection task, a place in the space can be chosen by the positioning task, an angle in the space by the orientation task, a number by the quantifying task, a sequence of characters among a special set of entities by the text task. But none of these interaction tasks modifies the objects directly. It is the basic purpose of another set of interaction tasks: continuous modification. These are “controlling tasks”. Their purpose is to form and transform visible objects. They characteristically control something, rather than specifying something. They are named according to the type of modification that they effect on the object:

- . stretch
- . sketch
- . manipulate
- . shape.

3.5.1. Stretch

A user takes a target object, moves it to new position distorting its shape by forcing one of its points to be agree with the position. Foley et. al. define typical stretching techniques as:

- . stretched lines.
- . stretched horizontal and vertical lines.
- . stretched vertices (lines possessing a common end point).
- . horizontal-vertical connections (called a zigzag).
- . stretched polygons, prisms and pyramidal forms (Foley et. al., 1990).

The interaction techniques and the application requirements of positioning task are similar with those of the stretching task. In particular, stretching techniques can be performed with continuous or discrete feedback, they can be direct or indirect and be used in two or three dimensions. The choice of the form of the object which will be stretched and the type of stretching differ also from positioning techniques. We can give more detailed information about typical

stretching techniques -such as extending a line from a fixed point to a specified point, stretching a line horizontally or vertically, drawing a number of rubber-banded lines, stretching a rectangle, expanding a circle or drawing a three dimensional figure such as rubber pyramid- are shown in figure 3.6.

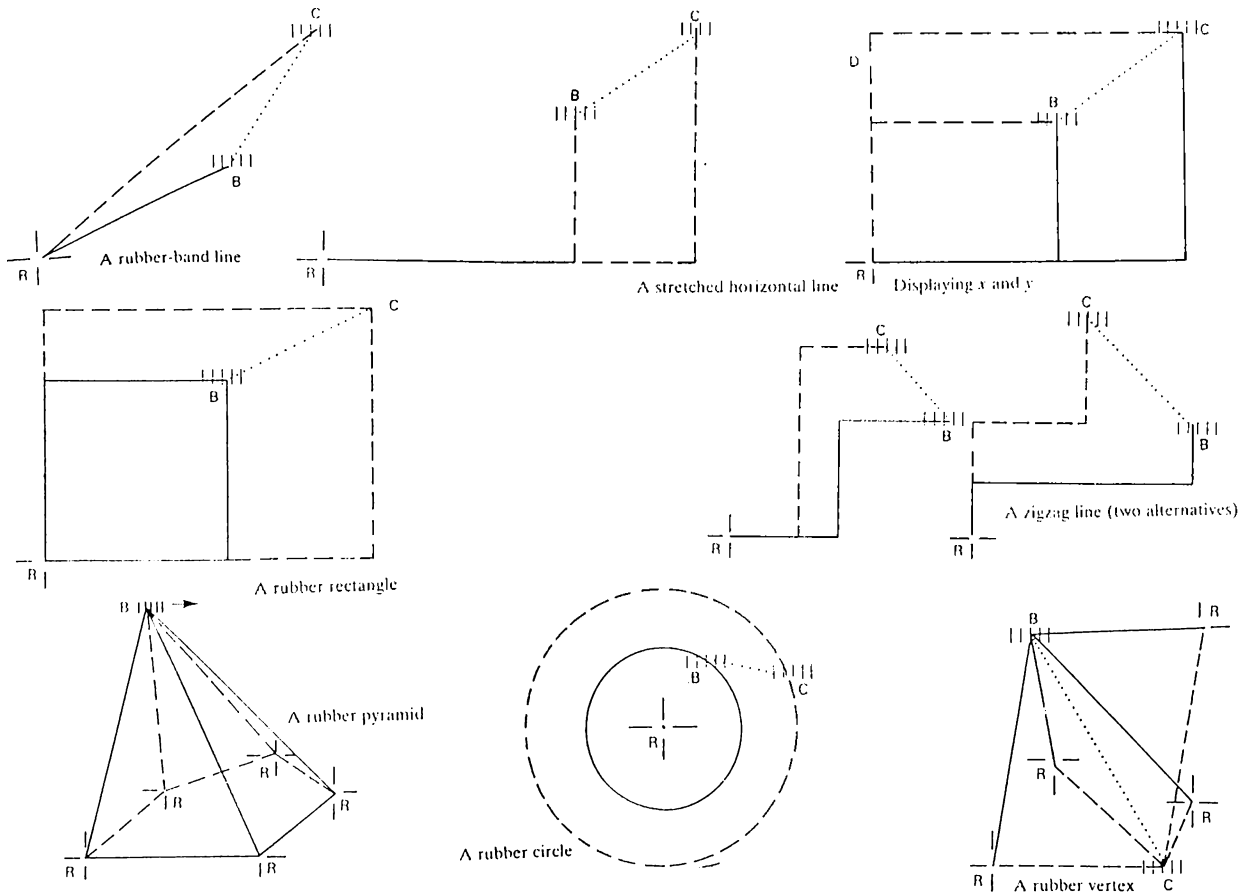


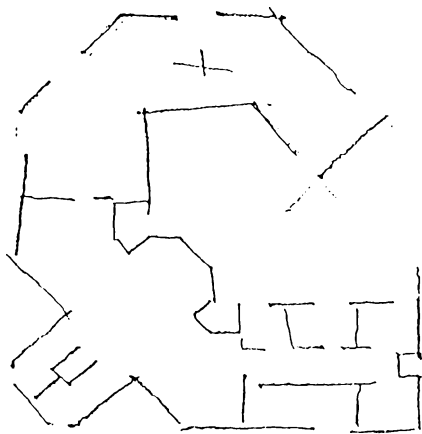
Figure 3.6. Typical stretching techniques (Foley et. al., 1990)

3.5.2. Sketch

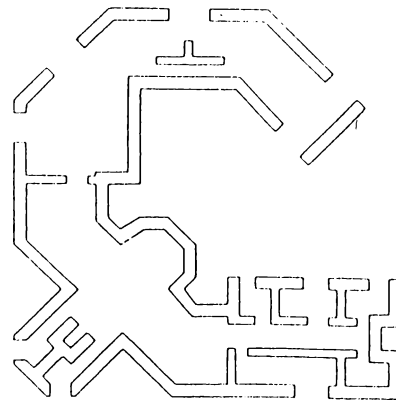
Sketching is a form of communication suited to architectural research, because architects do much of their work graphically. An architect sketches for two purposes: first, to convey to other people information that is difficult to express verbally, and second, to act as a sort of external memory, or in a sense to convey information to himself. Once the sketch has been committed to paper, he can modify it to change the information it contains. Changes can be prompted either by the decrease and flow of dialogue with an observer or by a change in the architect's own idea, brought about by the feedback loop running through brain, hand, paper, and eye. In either case, the sketch is important because of the intended meanings it contains. In a similar way, a computer system will be

useful as a sketching tool to be able to attach some meaning to the objects being sketched and creating an object by freehand sketching, by manipulating a locating device as it were a pen.

Durgun and Özgüç specify the main purpose of architectural sketch recognition as “understanding the intention of the architect from these rapid and unconstrained sketches” (Figure 3.7.) (Durgun and Özgüç, 1990). For this reason, the computer has to be knowledgeable enough about the subject matter being sketched to be able to ask some questions, and perhaps to offer some information of its own. So, the computer should be able to enter into a dialogue with the user. To provide it, the user has to specify a starting position, a path and an end. In this case, the requirements of this task are dimensionality, resolution, sampling criterion and smoothing method. Since the technique has a continuous-feedback, all the requirements for positioning task can be used. In addition it is also similar to the pathing task.



Freehand sketch prepared as an input.



Computer recognition of Figure

Figure 3.7. Computer recognition of a freehand sketch prepared as an input (Durgun and Özgüç, 1990)

In sketching, the time sampling -that means speed at which the user draws-, the pressure sampling and the space sampling are important requirements which will facilitate determining the user’s graphical intentions. The drawing speed of the user reflects his degree of purposefulness and his interest. It is usually true that when a person is drawing quickly, he is not so interested in detail as when he is drawing slowly. In a quick sketch, the person is usually interested in the general impression of the lines, rather than in the exact reproduction of those lines. Conversely, a slowly drawn sketch is often more detailed. In this case, the position of each line becomes important, and the sketcher wants his drawing to be seen exactly as drawn.

Another research that consists of 85 architects, made by Durgun and Özgüç, to find the determinant and graphical intentions, concludes the parameters to be considered as follows: the type of sequence in which the sketch is drawn, the most often used and predominant elements of a sketch -such as lines-, line conditions -such as straight lines, parallel lines, intersecting lines and curves-, corner conditions, distinction of openings from the unintended and discontinued lines (Durgun and Özgüç, 1990).

Foley et. al. add another requirement of the task as “approximation”. Its manner includes a specification of whether it is an exact matching or a smoothed approximation. In a smoothed approximation, the sampling points are used as control points (Foley et. al., 1990).

3.5.3. Manipulate

Changing the position and orientation of an object, without changing its form, can be made by a manipulation technique. It is carried out by applying appropriate geometric transformations to the coordinate points of a displayed object. The basic transformations can be defined as translation, rotation, scaling and reflection.

. A translation is a straight-line movement of an object from one position to another. A user picks or locates an object on the screen, adds the translation distances to the coordinates of the object or moves it to a new location; so a reference point on the object change with the other specified point (Figure 3.8).

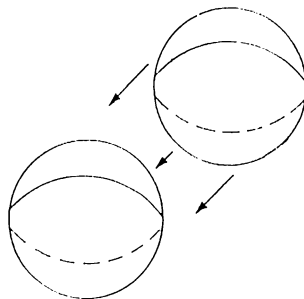


Figure 3.8. Translation of an object (Foley et. al., 1990)

. A rotation occurs with the transformation of displayed object's points along a circular path. This path can be specified by an axis and an angle which

determines the amount of rotation for each vertex of a polygon. The movement is normally continuous (Figure 3.9.).

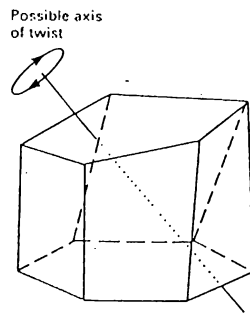


Figure 3.9. Rotation of an object (Foley et. al., 1990)

. A scale defines the size of an object displayed on the screen. The larger or the smaller appearance of the object on the screen can be manipulated by changing the scale. It is carried out by multiplying the coordinate values of each boundary vertex by scaling factors (Figure 3.10).

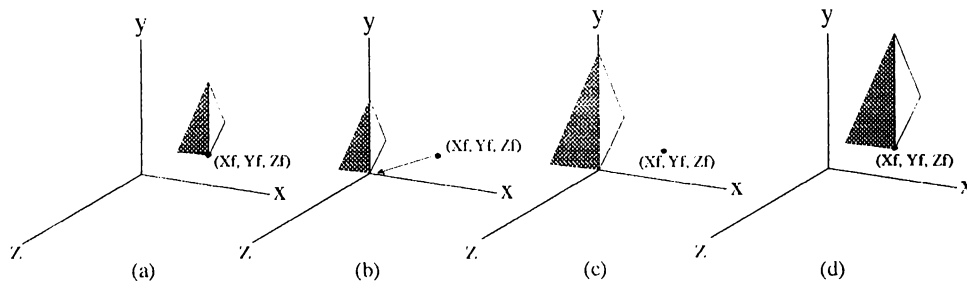


Figure 3.10. Scaling an object (Hearn, Baker, 1989)

. A reflection is a transformation that produces a mirror image of the displayed object. This image can be generated relatively to an axis of reflection (Figure 3.11).

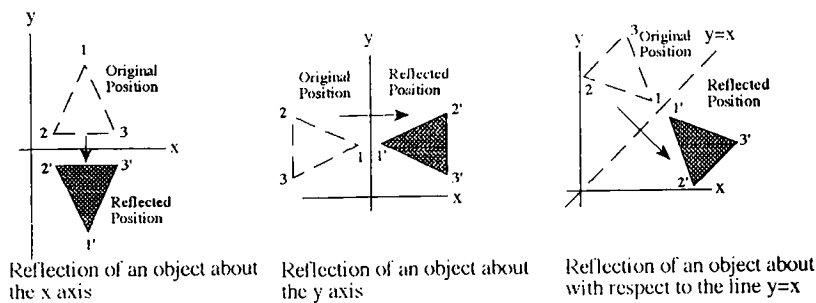


Figure 3.11. Reflection of an object (Hearn, Baker, 1989)

3.5.4. Shape

An object can be reached at the desired form by a shaping technique. This technique is dependent on how lines and surfaces are represented inside the system. A particular shape can be represented by control points. Two shaping techniques which use control points are the Bézier method and the spline method. They can be used to represent the complex curved lines in two or three dimensions. In the Bézier representations the control points are external to the curve, while in spline representations they lie on the curve. These methods can also be used to represent the surfaces by taking the Cartesian product of two curves which will represent the cross-section of the surface. To select and drag displayed control points to a new position using a locator is the most common technique for forming or reshaping curved lines and surfaces.

4. INTERACTION TECHNIQUES

The notion of interactive representation which is a communication system offers to the users some choices. The users have to think about the principles for the selection of materials and tools included in that representation. Those materials and tools will be shared both by the system and the user during the interaction. Users have also to consider how well these systems support the different stage of design. These are action specification and execution stages which can be done by a command language, or by pointing at menu options or icons, or communicating with speech or direct manipulation, or with another interaction technique. Each of these techniques has advantages and disadvantages that depend on several factors. Gaines distinguishes these different techniques by three main style of dialogue (Gaines and Mildred, 1986b):

- . Formal Dialogue: the activities and data structures are presented externally in a direct representation of the computer. Examples to formal dialogue are job control languages, simple prompt-response systems, menus, form filling systems and command driven systems.
- . Natural Language Dialogue: man uses the *language* to communicate information and commands. The dialogue is simulated within the context of the activities and data structures within the computer. Natural language is attractive because users already use it to communicate with other people.
- . Graphic Dialogue: man manipulates the *objects* to communicate information and commands. Its applications are in the form of light pens, touch screens, windows and icons.

Another categorization of human-computer interaction is given by Moran:

- . Application: Text Editing System
Line Drawing Systems
Computer Aided Design Systems
Computer Assisted Instruction Systems
Quality Assurance Systems
Process Control Systems
- . Dialogue Style: Function Key Systems
Menu Systems
Form-Filling Systems
Answer-the-Question Systems
Mixed-Initiative Systems
- . Language Type: Command Language Systems
Programming Language Systems
Natural Language Systems (Moran, 1981)

Baecker organizes the presentation of various techniques in terms of nine major general categories of interaction technique:

1. Command line dialogues
2. Programming language dialogues
3. Natural language interfaces
4. Menu systems
5. Form-filling dialogues
6. Iconic interfaces
7. Window systems
8. Direct manipulation
9. Graphical interaction (Baecker and Buxton, 1987)

We can say that there is no *best* overall interaction technique. All techniques have their specific *pros* and *cons* when different user groups, tasks and application areas are taken into consideration.

For example the peculiarities of command language interfaces -such as supporting user initiative, creating user defined macros and flexibility- may be an advantage for expert users; but also may be an errorless environment for novice users. The properties of menu-driven interfaces -such as reducing keystrokes, structuring decision making, permitting use of dialogue management tools- may be profitable for some tasks; but may also slow frequent users and consume screen space for some tasks where spacing is an important factor. The benefit of simplifying data entry when working with form-fill-in technique may be an handicap -because of consuming screen space- for some task types. Presenting task concepts usually in a direct manipulation may be hard for some users. Natural language interaction usually provides little context for identifying the next command and frequently requires clarification

dialogue; and this may be slower than the alternatives. But still, if users are knowledgeable about a task domain with a limited scope and their lore inhibit command language training, there exist opportunities for natural language interfaces. For some purposes, graphs, pictures and moving images will be significant and superior to words; in other situation words will be superior.

In conclusion the various kinds of requirements of different tasks can be responded by different interaction techniques. In other words, different kinds of interaction techniques have different properties which will be advantageous for some users or some tasks and disadvantageous for another. Therefore the dialogue should be designed to complement the task and the user. It should also allow both the user and the application to develop within the system structure and ensure not only the functionality but also usability.

For these aims the subtitles of the fourth section give detailed information about different types of interaction techniques and will be helpful for the selection of one of them for a particular task.

4.1. Command Language Interface

A command language interface is a system where the user enters instructions using well-specified and restricted grammar and vocabulary. This language may consist of single commands or have complex syntax. It may have only a few operations or a large number. Commands may have a hierarchical structure or permit possibilities to form variations. A typical form may be a verb followed by a noun and its qualifiers and arguments. Abbreviations may be permitted. Feedback may be generated for error messages. It may offer the user brief prompts or be close to menu selection systems. Finally natural language can be considered as a complex form of it. Figure 4.1. shows an example of a command language interface. This section will try to bring some clarifications to these options.

Shneiderman states the basic goals of language design as:

- . precision
- . compactness
- . ease of writing and reading
- . speed in learning
- . simplicity to reduce errors
- . ease of retention over time

And the higher level goals as:

- . a close correspondence between reality and the notation
- . convenience in carrying out manipulations relevant to the user's tasks
- . compatibility with existing notations
- . flexibility to accommodate novice and expert users
- . expressiveness to encourage creativity and
- . visual appeal (Shneiderman, 1987).

For these goals an effective command language must not only represent the user's tasks and satisfy the human needs for communication; but must also integrate the recording, manipulating and displaying mechanisms of the language in the computer.

```
-----  
|                   COMMAND OPTIONS_                   |  
| (Type 'help' for further details)                   |  
| |                                                     | |
| | append      cancel      end       fetch          |  
| | front      insert     join      prefix          |  
| | rubout     send       split                      |  
| |-----|                                           |  
| |                   TARGET SENTENCE                   |  
| | a stitch in time saves nine                       |  
| |-----|                                           |  
| | sti tch in a time xpqy aves nine                 |  
| | sti tch in a time aves nine                     |  
| | > a sti tch in time aves nine <                   |  
| | |                                                 |  
| |-----|                                           |  
| |                   _?                   |  
| |-----|                                           |  
| |                   Type your next command:         |  
| | join_                                             |  
| |-----|                                           |
```

Figure 4.1. A sample of command language interface (Hammond and Barnard, 1984)

Command languages are distinguished from other systems by their devices and information. For example they are distinguished from menu selection systems (which will be examined in detailed in section 4.3.) by the fact that the users of command languages must recall notation and initiate the action. Menu selection users receive instructions and choose one of them from a limited set of alternatives; they respond to an action more than initiating it.

Shneiderman determines the first step for the designer as “functionality” and the common design error as “excess functionality” and “insufficient functionality”. For these problems, he suggests the “transition diagrams” showing how each command takes the user to another state, and “macro facility” which can be programmed and include specification of arguments, conditionals, iteration, integers, strings, screen manipulation, plus library and editing tools (Shneiderman, 1987).

After having adopted a concept and a model for operations, the designer must choose a strategy for the command structure. Each command may be in a simple form to carry out a single task. For this case the number of commands will be equal to the number of tasks. With a small number of tasks this system will be simple to learn and use; but with a large number of commands, there will be the danger of confusion. In another situation a command may have one or more arguments indicating the objects manipulated or in another, they may have options indicating special cases. The arguments may also have options. But if the number of options augments, the complexity and error rate will increase. For a different situation all commands may be organized into a structure, like a menu tree. The first level might be the command action, the second one the object argument and the third one the destination argument.

A meaningful command structure will facilitate the human's learning, problem solving, and retention over time. If command languages are well-designed, users can recognize the structure and put it in their semantic knowledge storage. Command languages should firstly give the possibility to express the simple, familiar or well-understood features and secondly consider the more varying aspects. For this aim, using consistent argument positions is favored than the consistent direct object positions (Shneiderman, 1987).

On the other hand, Carroll (1982) suggests the organization of the names into paradigms incorporated with the concept of "congruency" and "hierarchicalness". The words chosen should reflect the functional relation between the name and the concept. In addition, the positional consistency and the grammatical consistency are important factors for recognition.

It is also shown that mnemonic names facilitate the comprehension better than the arbitrary names; so it is evident that specific names are slightly better than general names. They may be more descriptive, more distinctive; so more memorable. But in some cases general terms may be more familiar; therefore easier to accept. A syntax employing both familiar and descriptive with everyday words will provide a language that can be easily and effectively used. But it is also known that the concept of "naturalness" differs enormously from one individual to another (Baecker and Buxton, 1987).

Even though command names should be meaningful for human learning, problem-solving and retention, they must also be in harmony with the mechanism for expressing the commands to the computer. Shneiderman states the traditional and widely used command entry mechanism as the keyboard. This requires the brief and easy use of commands. In this case abbreviations

become attractive and necessary. Several strategies support that the abbreviation should be made by a consistent strategy.

The formal representation of command language syntax is also an important factor for the understanding of the user. Some interactive systems use special meta-characters to represent grammatical representations, someone use a form of prompt called command menus where users are shown a list of words and make a selection between them, (this type of interface will be examined in detailed in section 4.3.); or a graphic method (which will be examined in detailed in section 4.5.) or a hybrid system such as a menu-based system with command specification offered as an override facility.

4.2. Natural Language Interface

Natural language interface is the operation of computers by people using familiar natural language to give instructions. This interaction technique may increase the expressiveness of the user input and allow users to gain access to systems since they do not have to learn a command syntax nor select from menus. English is a common standard and is complete. It is natural besides being concise and no additional equipment is required. Transactions completed by a computer that understands English are cheaper, more accurate and more predictable than responses given by people. English does not grow old. Unlike a graphical interface whose novelty wears out, expressing questions in English is always a comfortable method of communicating. These are all advantages of an automated natural language interface (Miller and Walker, 1990).

Besides all these advantages there are also problems with this interaction technique such as implementation on the computer and desirability for large number of users for a wide variety of tasks. Natural language can be effective for the user who is knowledgeable about some task domain and computer concepts but who is an intermittent user who can not retain the syntactic details. Disadvantages of the technique include wordiness of natural language and the degree of coverage provided -a system that can understand only very limited part of English will not provide an effective natural interface. But people have not yet achieved an effective natural language communication with computers. So there are some factors be considered in deciding whether to use a natural-language interface. Baecker and Buxton (1987) states seven of them:

1. Cost of the interface: The natural language interface is more expensive compared with other more restricted interfaces with respect to its design and

implementation as well as the execution time.

2. Ease of learning: Since people already know a natural language they spend the minimum time and effort to learn such an interface.
3. Conciseness: Communication between the user and the computer depends on the number of keystrokes he must make.
4. Need for precision: Many English sentences stand for one more than one meaning so for programs where precision is important natural language interface may not be such adequate.
5. Need for pictures: With computer-aided design systems natural languages may be a good choice but with aid of graphic-based communication tools.
6. Semantic complexity: The power of English grows as the problem-solving and reasoning capabilities of target programs grow.
7. Promising more than can be delivered: If there is not a good match between a program and its interface, the close connection between the range and complexity of a program and the range and complexity of the interface to the program may cause problems.

As Shneiderman states:

Computers have impressive speed, storage and accuracy and unique input/output devices which are bypassed if we use natural language. Novel methods of pointing to, manipulating or changing displayed objects may be more appealing than lengthy and tedious natural language. Instead of building machines which mimic people, we need to develop an understanding of the distinct capabilities of computers and people (1987).

The technology of natural language processing: For a computer to interpret a relatively unrestricted natural language communication, a great deal of knowledge is required. Knowledge is needed of the structure of the sentence, the meaning of words, the morphology of words, the model of the beliefs of the sender, the rules of conversation, and an extensive shared body of general information about the world. Natural language communication between humans is very dependent upon shared knowledge, models of the world, models of the individuals they are communicating with, and the purposes or goals of the communication.

Many of the issues in natural language understanding center around the way people use language. Given speech acts can serve many purposes, depending on the goals, intentions and strategies of the speaker. Thus, methods for determining the underlying motivation of a speech act is a major issue. Another issue is understanding how humans process language.

Components of a natural-language understanding system: The process of translating statements from the language in which they are made into a program-specific form that causes appropriate actions to be performed consists of three parts:

- . Words and the lexicon: The first thing to do to understand a statement is to break it into its components; words. Dividing a sentence into words is lexical analysis.
- . Grammar and the structure of the sentence: In finding the meaning of a statement the first step is to assign to the statement a structure that will probably correspond in some way to the structure of its meaning. Assigning such a structure to an unstructured object is syntactic analysis.
- . Semantics and the meaning of sentences: Assigning a meaning to a statement for it to be understood is semantic processing (Baecker and Buxton, 1987).

Baecker and Buxton (1987) combine these components with three approaches:

Approach 1. Language through windows: If the number of statements the user needs to make to the target program is not large the system may display the available options to the user. The user chooses among these options and constructs a complete statement. Figure 4.2. shows an example of a screen presented to the user. A window-based, natural language systems runs efficiently because options available to the user are accurately constrained.

COMMANDS:	Find	Find the	Find all the
FEATURES:	CONNECTORS:	QUALIFIERS:	COMPARISONS:
part number	and	supplied by	between >=
part name	or	whose colors are	equal to <
quantity	of	whose price is	> <=
supplier name	the average	with shipment number	not equal to
supplier address	the lowest	whose name is	ATTRIBUTES:
supplier number	NOUNS:	whose address is	<part number>
price	parts	who supply	<quantity>
color	suppliers		<supplier>
	shipments		<price>
			<color>
QUERY SO FAR:			

Figure 4.2. A sample natural-language menu (Baecker and Buxton, 1987)

Approach 2. Semantic grammars: This is a straightforward extension of the window system to allow a greater number of user options. First by lexical analysis a statement is separated into words then words are analyzed for syntax and semantics in a single step. Semantic grammars are useful only when

a small subset of a language is to be recognized.

Approach 3. Syntactic grammars: In this approach both the input language and the program actions are examined and grammar rules that map as directly as possible are written. The rules thus appear semantic in that they relate directly to the target actions. But separating syntactic and semantic processing is necessary to make it possible to construct a grammar for a language once and to reuse it in many interfaces.

4.3. Menu-Driven Interface

A menu is a collection of selections displayed on the screen. The selections may be displayed in the form of text (usually words or short phrases) or icons. The user need only select an option to set the correct programs in motion. Menu selection is especially effective when users have little training and are unfamiliar with the terminology since they reduce the amount of information the user needs to remember. Menus provide an effective way to present a limited set of options to users. People use simple search strategies for ordinary menu sizes, are sensitive to menu length, and search easier with sorted menus.

Effective menu selection systems emerge only after careful consideration and testing of numerous design issues, such as semantic organization, menu system structure, the number and sequence of menu items, titling, prompting format, graphic layout and design, phrasing of menu items, display rates, response time, shortcuts through the menus for knowledgeable frequent users, availability of help, and the selection mechanism.

According to Shneiderman (1987), the primary goal for menu designers is to create a sensible, comprehensible, memorable, and convenient semantic organization relevant to user's tasks. Users should have a clear idea of what will happen when they make a choice. The user of a dialogue constructed with menus is faced with two basic problems: finding the item in the menu which he wants (this includes the problem of understanding different items in the menu) and knowing where he is in the system and thinking ahead to the next menu or menus. Menus have the advantage compared with yes-no questions in that the user has more alternatives to choose from, which gives more user power and fewer steps toward a desired user goal.

The simplest applications consists of a single menu; the second group includes a linear sequence of menu selections; strict tree structures make up the third

group; and acyclic (menus which are reachable by more than one path) and cyclic (menus with meaningful paths that allow users to repeat menus) networks make up the fourth group (Figure 4.3.) (Shneiderman, 1987).

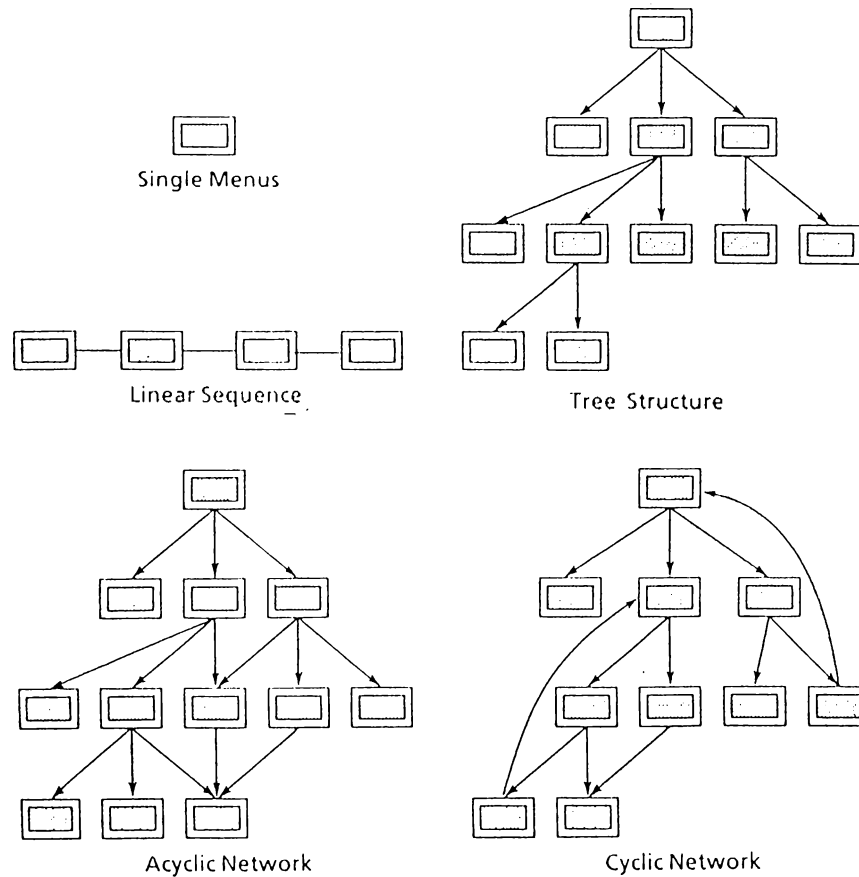


Figure 4.3. Different types of menu selection applications (Shneiderman, 1987)

Single menus may have two or more items, may require two or more screens or may allow multiple selections. Single menus may pop up on the current work area or may be permanently available. For these menus, a simple descriptive title that identifies the situation is all that is necessary.

The simplest case of single menu is a binary menu with yes/no or true/false choices. The menu items can be identified by a single letter which makes it more clear and memorable. If the single menu have more than two items then it is called the multiple item menu. If the list of menu items require more than one screen but allow only one meaningful item to be chosen then they make up the extended menu. The first portion of the menu is displayed with an additional menu item that leads to the next screen in the extended menu sequence. Another type, pop-up or pull-down menus appear on the screen in response to a click with a pointing device such as a mouse. Selection is made by moving the pointing

device over the menu items. Since the pop-up menu covers a portion of the screen the menu text is kept as small as possible (Figure 4.4.). Permanent menus, another type of single menus, can be used for permanently available commands that can be applied to a displayed object. A further variation on single menus is the capacity to make multiple selection from the choices offered.

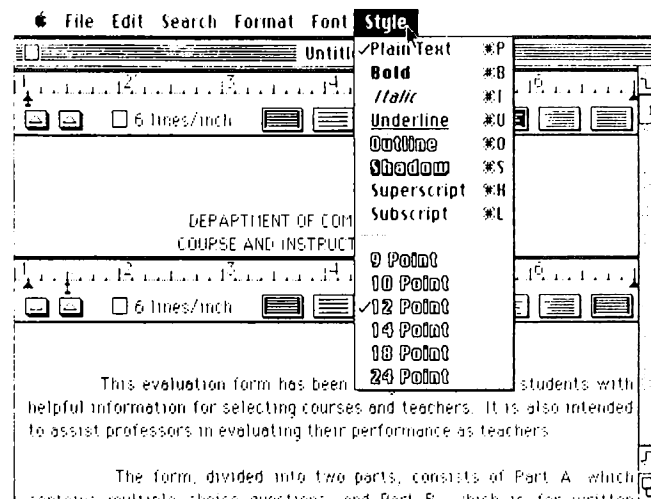


Figure 4.4. The pull-down menu on the Apple Macintosh MacWrite program (Shneiderman, 1987)

Linear sequence of menus are simple and effective for guiding the user through a decision-making process. The user should know the position within the sequence and have the ability to go back to earlier choices. With a linear sequence of menus, the titles should accurately represent the stages in the linear sequence (Figure 4.5.).

Tree structured menus have the power to make large collections of data. The depth (number of levels) of a menu tree item depends on the breath (number of items per level). If more items are put into the main menu, then the tree spreads out and has fewer levels. This seeing the full picture continuously aids decision-making. Kiger (1984) carried out a set of experiments in 1980 which aimed to look at the effect of the depth of a search on the response times and error rates at each level of the tree. He found that optimal menu interfaces take advantage of the limitation by pushing the upper limit on individual menus, and thereby reducing the depth requirements of the tree structure. As a general principle, he stated, the depth of a tree structure should be minimized by providing broad menus of up to eight or nine items each.

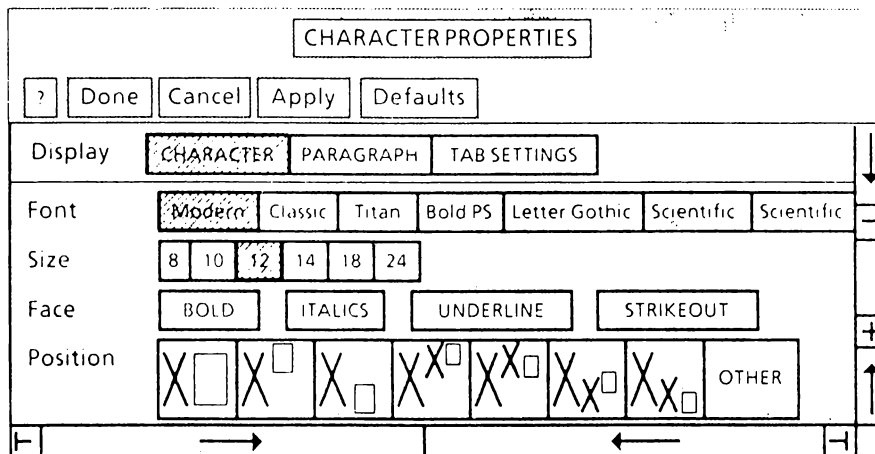


Figure 4.5. The linear sequence of menus on the Xerox Star (Shneiderman, 1987)

As Shneiderman suggests the rules for forming menu trees are:

- . Creating groups of logically similar items,
- . Forming groups that cover all possibilities,
- . Making sure that items are non overlapping.
- . Using familiar terminology, but making sure that items are distinctive from each other (1987).

Acyclic and cyclic menu networks are used when paths are permitted between disparate sections of a tree rather than user beginning a new search from the main menu.

Once the item in the menu have been chosen the designer is still faced with the problem of presentation sequence. In sequencing of items timing may be in chronological ordering; numeric ordering may be in ascending or descending order; physical properties like length, area, volume, temperature, weight, velocity may be in increasing or decreasing order.

A critical variable that effect users in menu selection is the speed at which he can move through the menus. This speed depends on the system response time, the time it takes for the system to begin displaying information in response to a user selection, and display rate, the rate in characters per second at which the menus are displayed. If the response time is long then menus with more items on each menu should be created to reduce the number of menus necessary. If the response time is long and the display rate is low then command language strategies become more interesting.

Shneiderman summarizes the menu selection guidelines as follows:

- . Use task semantics to organize menu structure (single, linear sequence, tree structure, acyclic networks, and cyclic networks)
- . Try to give position in organization by graphic design, numbering and titles
- . Items become titles in walking down a tree
- . Make meaningful groupings of items in a menu
- . Make meaningful sequences of items in a menu
- . Items should be brief and consistent in grammatic style
- . Permit type-ahead, jump-ahead, or other short-cuts
- . Permit jumps to previous and main menu
- . Use consistent layout and terminology
- . Consider novel selection mechanisms and devices
- . Consider response time and display rate impact
- . Consider screen size
- . Offer help facilities (Shneiderman, 1987).

Just because a system has menu choices written with English words, phrases, or sentences does not guarantee comprehensibility. In phrasing of menu items familiar and consistent terminology should be used, each item should be clearly distinguished from other items, items should ensure consistency and conciseness, and the first word should help the user in recognizing among items.

The constraints of screen width and length, display rate, character set, and highlighting techniques strongly influence the graphic layout of menus. But there are some consistent menu components. For example left justification is an acceptable approach for titles. Also items should be left justified with the item number or letter preceding the item description. The instructions, if identical in each menu and placed at the same position help the user more. Some systems indicate which portion of the menu structure is currently being searched, which page of the structure is currently being viewed, or which choices must be made to complete a task. This information and any other error message should appear in a consistent position.

So the designer in menu-driven interface should understand the semantic structure of his application. Then should concentrate on organizing the sequence of menus to match the user's tasks ensuring that each menu is a meaningful semantic unit. If some users make frequent use of the system, then typeahead, shortcut, or macro strategies should be allowed. Simple traversals to the previously displayed menu should be permitted. When the system is implemented the designer might collect usage data and error statistics to guide refinement. Commercial menu creation systems are available and should be used to reduce implementation time, ensure consistent layout and instructions, and simplify maintenance.

4.4. Iconic Interface

Iconic communication is an interaction technique which conveys ideas or information in a nonverbal manner, using *images*. The images related to the idea are chosen either by resemblance (pictograph), by analogy (symbol) or by the selection from a previously defined and learned group of arbitrarily designed images (signs) (Lodding, 1983). In fact, a unique discipline of *iconic communication* does not exist. For understanding the design, the application and the potential of an image, some diverse fields must be understood. Figure 4.6. indicates the scope of the iconic communications.

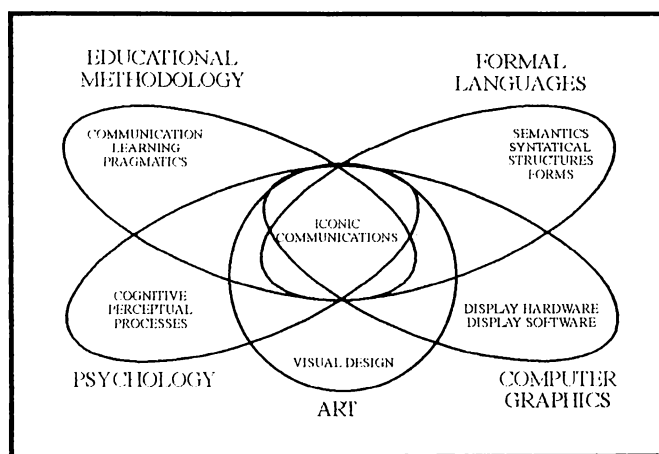


Figure 4.6. The scope of iconic communications (Huggins and Entwisle, 1974)

The graphic display provides a means to present final information results. In addition to this, the apparent complexity of the information systems tool itself can be reduced by a mechanism: by an iconic interface. The iconic form of programming uses special and object knowledge, and replace the linguistic means of words with logos. It can represent a lot of information in a small space and visually is more distinctive than a set of words. Commands and system information presented in the form of icons benefit the new capabilities of graphics displays, reduce the learning time and effort, and facilitate user performance while reducing errors. People have always found natural to communicate with images. Because, a person has to make a special effort to remember all the detail of a linguistic message. Generally when we *remember* something, it is restatement or a translation of the original. But an iconic interface requires only that the user *recognizes* an image. This recognition can be unclear; the user does not have to describe the image, nor name it; he only pick the icon to select. In addition the human mind has powerful image memory and processing capabilities, so he finds easier an image-based interface requiring to

recognize and point than a text-based interface requiring to *remember and type*. Besides these advantages, using icons has also the difficulties. The user finds an image clear and easy to understand when it possess a close resemblance to a particular object; in that case the number of expected messages is limited. But, in other situations, the user needs additional support to decode the message. Furthermore, the speed of image processing and the accuracy of image recognition are two important factors for iconic-based man-machine interface. Also the display unit with too low resolution, causes icons to be poorly represented, so replacing them with text should be considered. Lodding states the requirements for the correct interpretation of an image with three factors:

- . the image code
- . the caption, and
- . the context (Lodding, 1983)

The image code is the representation -the image itself. The caption is the additional lexical channel which reinforces and expands the picture message. And the context is a frame of reference which interprets the image and the caption.

From the designers' point of view, a unique difficulty for iconic communication is that they have no "dictionary" from which to select an appropriate image. They choose a representation that they try to give the intended information. If they essay to be too realistic, the details can slow down recognition and interpretation. In contrary a highly stylized design will be either not understood or interpreted in many different ways.

From the users' point of view, there are some difficulties about the acceptance of an icon. For example an image can convey certain undesirable messages or non-understandable language and cultural barriers. Because icon designs are influenced by a large number of factors, including the designer's cultural background, education and environment which can result a "nonuniversal" icon. There is also the problem of the confusion of images that are used to convey information. Arnheim (1969) suggests a taxonomy which will aid to order this confusion; and he identifies the starting point for developing an icon taxonomy as "understanding the functions supported by images". The terms used to define the usage of the image are "picture, symbol" and "sign". Picture reflect the relevant qualities of an object or an activity. A symbol presents a concept rather than a particular object. A sign functions as a reference to the object or concept. Arnheim (1969) classifies an icon by its design style, and pairs each of the image function with a design styles as follows:

Design:
 representational
 abstract
 arbitrary

Function:
 picture
 symbol
 sign

A “representational image” serves to a general class of objects. That means, it is typical. The advantage of this design style is to recognize easily objects referred to it. They can also taught, learned and retained easily. The major difficulty of this design style is the change of form of the most technical/cultural objects over time. Figure 4.7. shows representational pictographic/object based icons from the Apple Macintosh system.

“Abstract icons” presents a concept to the viewer who is away from the concrete image. The intention is not to show the object. It is to convey a concept at a higher level of abstraction than the symbol itself. In an abstract design the image is reduced to its essential elements. Because the designer is attempted to focus on a particular concept rather than the object itself. Figure 4.8. shows an abstract/symbol icon.

Consequently, when the purpose of an icon is not representation, an “arbitrary icon” can be “invented” and assigned a meaning. When it does not aimed to tie the intended message to an object, arbitrary icons can be used. Figure 4.9. shows an arbitrary/sign icon supported by caption.

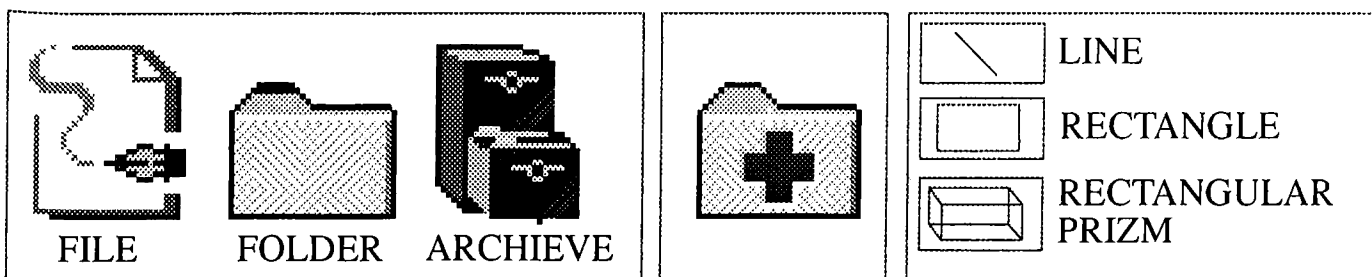


Figure 4.7.
 Object- based objects
 currently found on
 the Apple Macintosh
 system

Figure 4.8.
 An abstract/
 symbol icon for
 “utilities”

Figure 4.9.
 Drawing icons carrying a
 relational structure similar
 to the one found in the
 line/rectangle/rectangular
 prizm relationship

Lodding divides the design process of icons into three different steps (Lodding, 1983) :

1. stating the message,
2. rendering the design, and
3. testing the resulting icon.

Identifying and defining the initial icon design in terms of the caption and the context points is the first step in the icon design process. Another and the most difficult part of the process is choosing one of the design style among representational, abstract or arbitrary types for the image. For the second - rendering the design- stage of the process considerations must be countered: image specific or image grouping. A well-designed icon should have simple shapes, grouped elements, and distinct separation of figure from ground. Color and spatial distortion are also two another factors that must be considered. The solution for color distortion is never allow color to carry information in the icon; and for spatial distortion is to avoid the use of extremely complex lines and shapes. For testing the resulting icon some several factors are :

- . being able to infer the intended meaning at the first view of the user.
- . appearing only appropriate icons for one selection.
- . not conveying any unnecessary negative connotations (Hersh, 1982) .

While the single-view single-icon model has proven very intuitive and easy to use, some objects have more than one logical view. (Draper, 1986) Multidimensional icons group set of icons, each describing a unique view of an object, into a single entity. The individual icons are located onto the sides of a simulated cube. Henry and Hudson argue two distinct advantages of using a cube instead of displaying all of the icons in a menu: The first one is that cubes are very familiar objects, and for a natural mental model the faces of a cube can be thought as the view of the entire cube, that means of the actual object. The second advantage is that the rotation of the cube which shows the accessible icons, uses only a small part of the screen space. (Henry and Hudson, 1990) Figure 4.10. shows an example of a multidimensional icon which represents a file and its five distinct views.

Since only three of them are visible at any one time, rotating the cube allows the selection of hidden faces. The way the user interacts with a multidimensional icon is illustrated in figure 4.11.

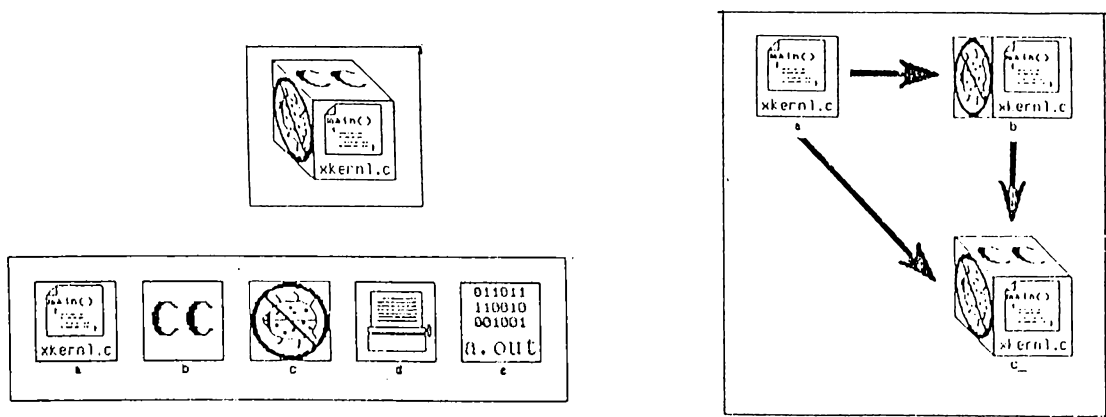


Figure 4.10. an example of a multidimensional icon which represents a file and its five distinct views (Henry and Hudson, 1990)

Figure 4.11. The way user interacts with a multidimensional icon (Henry and Hudson, 1990)

The multidimensional icons group commands with objects, and they avoid the users applying commands to inappropriate objects. In addition, the cube analogy provides a reminder to the user forgetting hiding views and remember that there are some views on the back of the cube.

4.5. Graphical Interface

Graphic communication is a design language made up picture images that are used to compress and convey ideas of shape, size, and construction of parts or whole objects. This interaction is a set of actions of a computer graphics system and its user on each other. The user provides input to the system via a set of devices; the system provides output to the user via a set of displays. These inputs and outputs must be reciprocal, that is, they must be related to one another. Thus, graphical interaction is a succession of interrelated actions and reactions.

In order to build useful interfaces, some displays techniques and pointing devices are required. Salvendy (1984) argues that the naturalness of object-based interfaces is based on the experience and skill of the user that the developed when dealing with physical objects. Interfacing with dynamic physical objects allow the user to reasoning about the system by maintaining the information, depicting topology, and permitting direct manipulation.

In this case, instruments and tools will be two important metaphors that can define the desirable properties of interactive graphics systems. They can be instruments for expression or measurement, tools to cause effects, vehicles for exploration and media for communication. In figure 4.12. Some of the tools of a CAD software are shown in Apple Macintosh.

Ryan (1986) states that to have optimum value, the format of these graphics must be clear, concise and subject to one and only one interpretation. Even more important is the method chosen to produce those graphic images. Baecker and Marcus (1989) add ten fundamental principles, and seven secondary principles which allow to use effectively the elements of this visible language. They states the ten fundamentals as “legibility, readability, clarity, simplicity, economy, consistency, relationships, distinctiveness, emphasis and, focus and navigability”; secondaries as “page characteristics, page composition and layout, typographic vocabulary, typesetting, symbolism, color and texture, and metatext”.

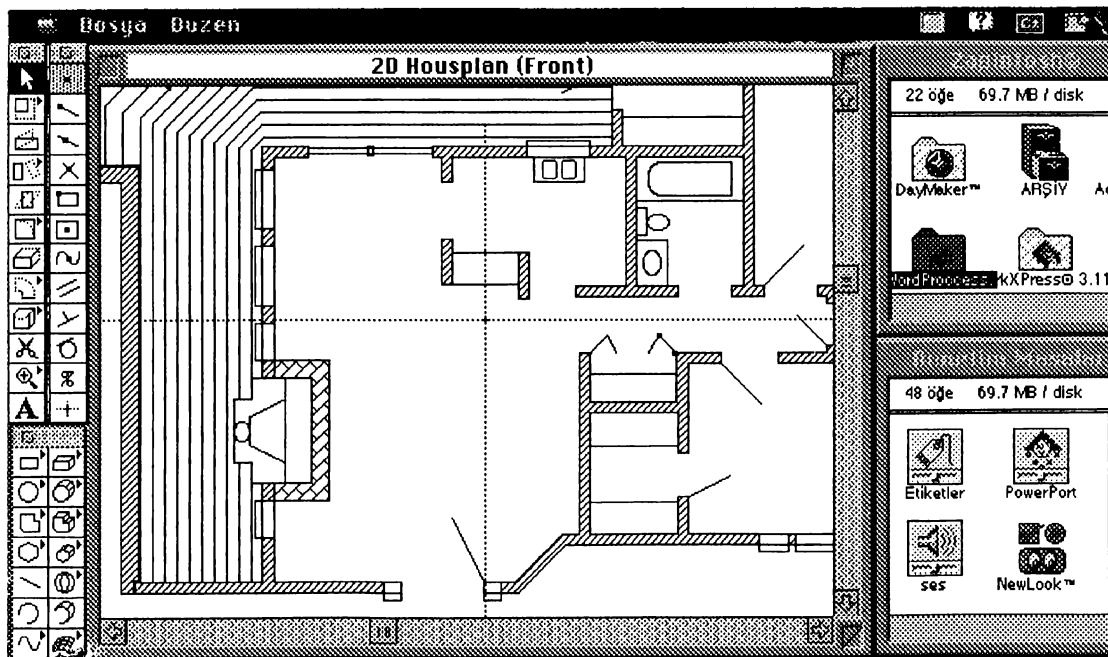


Figure 4.12. A CAD software on the Apple Macintosh

Baecker and Buxton categorize graphics systems, architecturally in four groups: Stand-alone single-user graphics machines, time-shared graphics systems, single-user graphics satellites, and time shared graphics satellite systems. In a

stand-alone single user machine, immediate feedback to graphical input can be achieved by calling input primitive with the code to generate output. Since all systems resources are available for this task, response time is only limited by the available computational bandwidth and the characteristics of the display device. However, in the other kind of graphics systems, there may be delays between the execution of the functions. This may be due to the competition of other users in time-sharing systems. To guarantee the integrity and responsiveness of feedback of interactive input, appropriate modifications, should be chosen (Beacker and Buxton, 1987).

Any graphical user interface is composed of two parts: the presentation or layout, which defines what pictures are on the screen, and the interaction and behavior, which determines how these pictures change with user actions. Baecker and Buxton (1987) distinguish inferencing in three different ways: First way infers how various objects in the scene are related graphically. When the designer draws an object, it usually has some relation with other objects that have already been drawn. For example, a box might be placed next to or inside another box. If the picture was simply a static background that never changed, it would not be important for the system to notice these relationships. The second type of inferencing is to try to guess when control structures are needed. For example, when the designer displays the first two elements of a list, the system infers that the entire list should be displayed and will generate an iteration. The final type of inferencing is to try to guess when actions should happen during the execution of an interaction. For example, a highlight bar might be displayed when the mouse button goes down.

One of the primary innovations of graphical interface is to allow the interaction portion of a user interface to be specified by demonstration. Just as what the end user *sees* is always visible to the designer, what the end user will *do* can also be executed at any time. The designer can either use the simulated or the real devices while in execution mode. Green (1979) sees these input devices as clustering into four categories, providing discrete or continuous data in single units or in sequences. A button box or an interval timer are devices that produce single discrete data items. A mouse, tablet, or light pen can produce single continuous data items if it is tracking or dragging and continuous sequences if it is inking, and single discrete data items if it is pointing or selecting (picking). In effect a set of seven virtual devices -a signal, a timer, a selector, a writer, a tracker, a dragger, and a inker- can be defined. A "signal" consists of the depression or release of a button or a contact switch or the change of state of a toggle switch. A "timer" consists of the provision of an alarm or signal after some period of time has passed. A "selector" consists of pointing to a particular

segment with a mouse, tablet, or light pen. A “writer” consists of a sequence of keystrokes on a keyboard. A “tracker” consists of a movement of the mouse or stylus terminated by a signal. A “dragger” is similar to a tracker except that the segment moved is one of the currently visible picture segments rather than a special tracking symbol. An “inker” consists of a movement by the mouse or stylus which results in a sequence of coordinate pairs. These positions are displayed by the appearance of an ink trail.

On the other hand Whitehead (1984) makes a distinction between input devices as direct or indirect devices. He argues that indirect devices (e.g. mouse, tablet, joystick), after some initial learning, allow a more comfortable operating position with extended use, get round the visual feedback difficulties of direct devices (e.g. light pen, touch screen) and have the facilities for signal transformations. The tablet has a particular advantage in its flexibility, since it can be used for selection or for digitizing or sketching.

In addition Ryan (1986) lists the items used in graphic communication workstations in table 4.1.

Traditional	CAD approach
drawing board	digitizing surface
t-square or parallel bar	coordinate measuring device
triangles	function keyboards
scales	factors
drafting machines	digital plotters
protractors/machine controls	subroutines
typewriters	keyboards for digitizers
drafting surface	graphics tablet
curves and templates	menu items
drafting aids	tablet accessories
pencil pens and knives	light pen, pencil, mouse
pencil pointers	pen functions, joysticks
erasers and shields	page, delete keys
sheet fasteners	electrostatic hold-downs
compass and dividers	image generators
lettering guides/devices	character generators
drawing paper	drawing paper/films
sketchpad/notebook	personal computer/accessories

Table 4.1. The items used in graphic communication workstations (Ryan, 1986)

4.6. Form Filling Interface

For the types of interaction tasks where many fields of data are necessary, “form fill-in” interaction technique may be an appropriate style. For this case the keyboard which can be viewed as a continuous single menu from which multiple

selections are made rapidly, may also be an appropriate device. For example, the user might be presented with a purchase order form for ordering from a catalog, as in figure 4.13.

Type in the information below,
 pressing TAB to move the cursor, and
 press ENTER when done.

Name: _____ Phone: (____) ____-____

Address: _____

City: _____ State: __ Zip Code: _____

Charge Number: ____ ____ ____ ____

Catalog Number	Quantity	Catalog Number	Quantity
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

Figure 4.13. A form fill-in design for a department store (Shneiderman, 1987)

The form filling approach is an attractive system. The user can see the full complement of information. This visibility gives the user a feeling of being in control of the dialogue. Some few instructions are required from the user where it is an approach resembling familiar to paper forms. But in some cases - especially for expert users- this locus of control which tend to be very much based within the computer may feel constraint and frustrate the user. For this reason some shortcuts should be provided where possible. The navigation should allow the user to move freely within the form but prevent him/her also from getting lost through the options. Form filling approach must be done on displays, not on hard copy devices. Consequently the display device must support cursor movement.

The types of form filling interaction may be *system driven* or *spreadsheet*. For the first type user inputs data in highly structured, system driven way. For the second, user is presented with a blank shell which can be filled with many types of data and options.

The primary factors which influence the quality of form filling interfaces are stated by Baecker as: The extent to which the logic of the form reflects the logic of the system structuring the input, the clarity of the design and the visual presentation of the screen, and the input form which facilitate the keying of data (Baecker and Buxton, 1987).

In addition Shneiderman explains the form fill-in guidelines as follows:

- . Meaningful guideline
- . Comprehensible instructions
- . Logical grouping and sequencing of fields
- . Visually appealing layout of the form
- . Familiar field labels
- . Consistent terminology and abbreviations
- . Visible space and boundaries for data entry fields
- . Convenient cursor movement
- . Error correction for individual characters and entire fields
- . Error messages for unacceptable values
- . Optional fields should be marked
- . Explanatory messages for fields
- . Completion signal (Shneiderman, 1987)

In addition to these, the designer should be alert to some special cases such as addition of extensions or the nonstandard formats.

4.7. Window-Oriented Interface

The graphical technique of defining a number of windows on a single display screen allows the user to see multiple sets of information at the same time. Windows are originally designed as explicit supports for the conduct of multiple activities. Window systems make possible the display of considerable information for each of the multiple activities that are currently active, subject to limitations on the size of the screen and the memory space allowed to handling the screen map (Figure 4.14.). They can be enlarged or shrunk to an appropriate size, moved around the screen and overlaid upon one another. All sorts of reminders can be presented on the screen because a major portion of them are continually visible. Windows themselves can serve as reminders of the existence of the activities contained within them. Windows, icons, or other reminders should have fixed positions, and each time the computer system is used, the same position is always used for the same information.

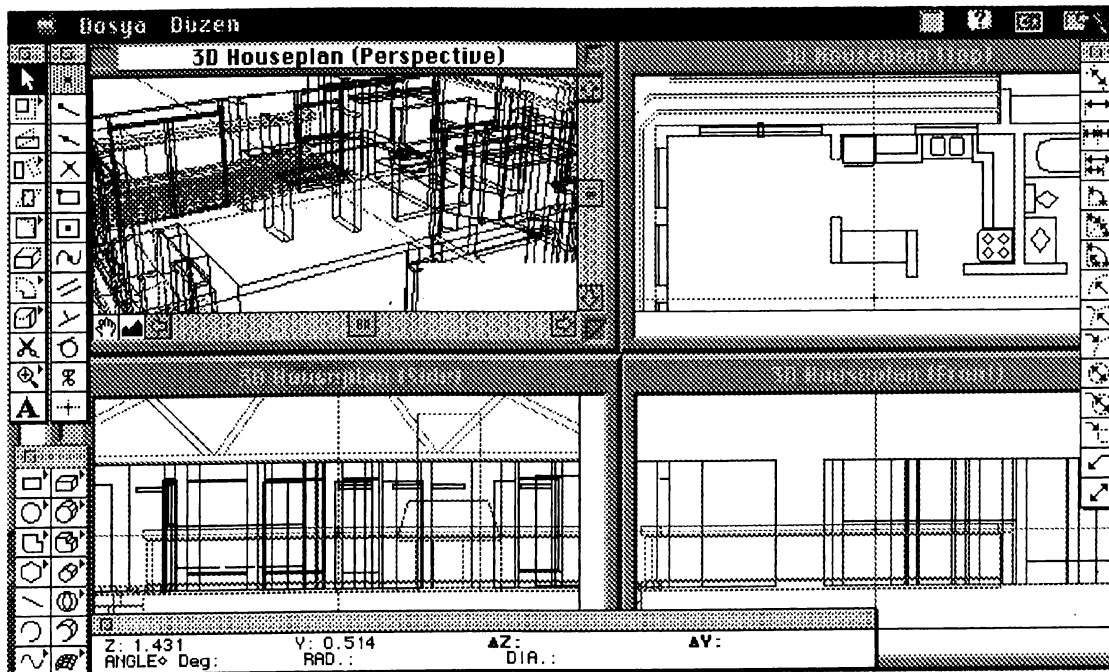


Figure 4.14. A window-oriented interface from a software of the Apple Macintosh

As Card et al. indicate seven task needs and types to which multiple windows could suitably be applied can be summarized as follows:

1. Fitting large amounts of information onto the screen (using overlapping or compressed windows).
2. Gaining access to multiple sources of information (one source per window).
3. Combining multiple sources of information.
4. Independently controlling multiple programs.
5. Keeping track of information likely to be used in the near future.
6. Setting the context for a set of commands.
7. Presenting multiple representations of the same task (Card et al. 1984).

Some different current designs for window systems stated by Baecker and Buxton (1987) are as follows:

Time-multiplexed windows: These type of windows can come up in two different forms; scrolling windows and frame at-a-time systems. Scrolling windows are often used with text-editors. The user edits his text in a window, but has available commands that can cause the text to move up, down or to a certain place, as if the user had a movable window he could position in front of a long

scroll. In frame-at-a-time systems while using a menu the user slips back and forth among a number of frames but only one frame is visible at a time.

Space-multiplexed windows: This type of windows can be one-dimensional, two-dimensional, two-and-a-half dimensional, or split vs. independent forms which are classified with respect to them divided into separate windows with different number of dimensions; their placement while being overlaid by the others; or one being split into smaller ones with carrying closely related information.

Icons: These are very small windows, generally represented on the screen by a small symbolic picture of some sort.

Bifocal windows: In this type, information is organized hierarchically in full detail in the center.

Optical fish-eye windows: Information in the window is compressed like the image of a convex mirror.

Logical fish-eye windows: Information detail may be reduced according to its logical distance from some focal point.

Zooming window: Data in the window or the window itself gets larger or smaller in the manner of a zooming camera.

Reichman states three forms of interaction: indirect, limited direct, and direct. The screen is the visual interface to the computer, it echos back the user the commands being issued to the computer and it reflects the computer's responses to these commands. This is the indirect interaction since there is not a direct, real time interaction between the user and the computer. While communication between the user and the computer is not direct as a result of their sharing a fuller and richer environment, the communication is still limited and at any point the user is only actively engaged in a single process; this is then the limited direct interaction. Windows divide one screen into multiple virtual screen, each behaving like a complete one where within each a different process can be carried out. So it is composed of a whole set of interactive processes, many of which are simultaneously visible to the user where direct interaction occurs (Reichman, 1986).

Windows provide us with a visual display of contextualization. In particular, there is no differentiation in activity status between the different nonactive windows in the environment. Because the window systems give visual evidence

of contextualization, users often assume that the conventions of contextualization used in everyday interaction are also supported by these systems.

A main feature of context support is supporting the relations between objects within a single or multiple context. Supporting context entails two things: knowing when things should be interpreted together and knowing when they should be interpreted separately. Context switching and interleaving are basic features of human interaction, whether in ordinary everyday conversation, using computer databases, main frame editors, or personal machine facilities (Reichman, 1986).

Not all windows are functionally independent. The language of communication in multi-window systems should include primitives for specifying the interrelation between the different context that user is setting up in the different windows. The problem of context visualization and support extends beyond just the windows. The problem is systematic to all graphical objects on the display. Basically, the user needs a visual constraint language for display objects, whether they are entire window contexts or particular entities within these contexts.

A differentiation between the status of different context/activities is important in the window-system. Open contexts do not have to remain on the display while the interrupting activity is executed; controlling contexts in contrast do and thus should be left on the display automatically by the system. A context is either active or nonactive and differential access to them, in general, is not supported via functional or structural markers.

To support user needs:

1. There should be underlying support to define individual contexts and the relations between these contexts and the objects they contain.
2. These dependencies and interrelationships should be made explicit on the visual display.
3. There should be a separate language that allows users to note the types of activity shifts being made.
4. There should be a correspondence between contexts and windows. The semantic import of a window should be that it constitutes a context. Linking contexts then becomes the equivalent of linking the visual reflection of these contexts, that is, windows.
5. The constraints derived from a context's status and interrelationship with

other windows should be self-evident to the user.

6. Navigational schemes should be provided to users so that they can navigate back to old contexts via reference to a functional relation that this proceedings context has with a current one (Reichman, 1986).

The displays should reflect status assignment. A minimum of four categories are required:

1. active
2. controlling
3. generating
4. closed (Reichman, 1986).

Only windows related to the current development of the active window would be visible at any given time.

A set of activity interrelationships should be defined. A set of object interrelationships should be defined. There should have a dynamic, easy, and nondestructive means for users to communicate these types of interrelationships to the computer. The computer should then visually reflect these relations back to users and provide them with a set of varying navigation mechanism which are derived from, and are based on, the different types of relations involved.

4.8. Direct Manipulation

The promise of direct manipulation is that instead of an abstract computational medium, all the “programming” is done graphically, in a form that matches the way one thinks about the problem. The desired operations are done simply by moving the appropriate icons onto screen and connecting them together. Connecting the icons is the equivalent of writing a program or calling on a set of statistical subroutines, but with the advantage of being able to directly manipulate and interact with the data and the connections. There are no hidden operations, no syntax or command names to learn. So it is an interface style in which the user can point at a visual representation of the task, manipulate it and immediately observe the results and is in control of the interaction. What you see is what you get (WYSIWYG). Shneiderman has suggested that direct manipulation system have the following features:

1. Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
2. Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.
3. Knowledgeable intermittent users can retain operational concepts.
4. Error messages are rarely needed.
5. Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.
6. Users have reduced anxiety because the system is comprehensible and because actions are so easily reversible (Shneiderman, 1982).

The term “direct manipulation” was coined by Shneiderman to refer to interfaces having the following properties:

1. Continuous representation of the object of interest.
2. Physical actions or labeled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible (Shneiderman, 1982).

A special type of direct manipulation, called WIMPS which stands for Windows, Icons, Mouse, and Pull-down menus, is typified by Apple Macintosh desktop. (Figure 4.15.) Objects such as applications, documents, files and drawings are represented as icons which the user can address with a mouse -controlled pointer. Pointing and selection invoke a system operation such as opening a document for word processing. The objects can also be moved or dragged around the screen.

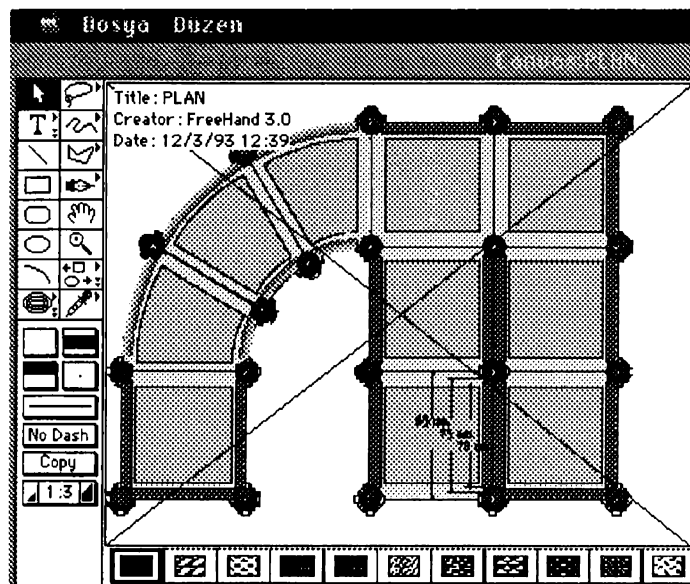


Figure 4.15. An Apple Macintosh software that offers direct manipulation

The main advantage of direct manipulation systems is in ease of learning and ease of use. If mapping is done correctly, then both the form and the meaning of commands is easier to acquire and retain. Interpretation of the output is immediate and straightforward. Direct manipulation provides a far easier means of constructing a drawing in architecture than by entering coordinate values through the keyboard. It can also be used as support for other interaction techniques like selecting menu options, pointing to function keys and buttons. It has the power to attract users because it is comprehensible, natural, rapid, and even enjoyable. Since it is easy to learn and use, it retains over time. Actions are rapid, incremental, reversible, and often performed with physical actions instead of complex syntactic forms. The results of operations are immediately visible, and error messages are needed less often, where is a big relationship of what is done and seen by the user, and the effect of the operation on the inner state of the system. Modeling direct manipulation requires understanding of the relationship between key and command, state and display. Hence, pressing the delete key deletes immediately; mouse movement moves the pointer; join clicking invokes a function that corresponds in some sense to the meaning suggested by the icon.

One of the problems with direct manipulation is that use of spatial or visual representations is not necessarily an improvement. The content of graphic representations is a critical determinant of utility. The wrong information, or a too cluttered presentation, can lead to greater confusion. A second problem is that users must learn the meaning of components of the graphic representation. Another problem is that the graphic representation may be misleading. The user may rapidly grasp the analogical representation but then make incorrect conclusions about permissible actions. Also graphic representations make the excessive screen display space. Another problem is that for experienced typists, moving a mouse or raising a finger to point may sometimes be slower than typing.

Direct manipulation interfaces have difficulty in handling variables, or distinguishing the depiction of an individual element from a representation of a set or class of elements. Direct manipulation interfaces have problems with accuracy, for the notion of mimetic action puts the responsibility that is often best handled through the intelligence of the system, and sometimes best communicated symbolically.

4.9. Speech Communication

Speech is an efficient and convenient vehicle for communication, being fast, universal and resistant to mispresentations. Speech also has undisputed advantages in certain situations where other media are impossible or inconvenient to use, and for particular classes of users. Speech is useful primarily for complex tasks requiring cognitive and visual effort, whereas simple tasks involving the copying of numeric data is carried out more quickly and accurately with keyboard entry as compared with voice entry. The benefit to be delivered from voice input and output is highly dependent on the specific task and environment. The selection of tasks for speech recognition should be based on specific task requirements. Speech is not a useful substitute for manual data entry when such tasks are already being performed successfully. Speech input is to improve system throughout only the complex tasks that involve high cognitive, visual, and manual loading.

The design of task-and-hear interface is not like the design of conventional type-and-see interfaces. One of the four reasons for this is the structural properties of the speech medium. In the usual type-and-see interface, an output to the machine stays on the screen until the user takes some action to dismiss it. The users deal with outputs similarly. Speech, on the other hand is transitory, once said it is gone. The second reason is short coming of today's speech technology. People can separate meaningful speech from noises, machines cannot. Pronunciation varies as a function of talker, rate of speech, and other factors; machines do not have such specifications. The third reason is the human as speech producer and perceiver. Successful recognition rate increases after first weeks as if machines were training the user. On the output side, the main factor is the limit of short-term memory. Finally the fourth reason why talk-and-hear interface is not like type-and-see interface is the kinds of tasks that speech is asked to do. Tasks that talk-and-hear interfaces do well are ones that involve simple data input, usually with hands and eye busy (Salvendy, 1984).

Speech recognition system is composed of a human speaker, a recognition algorithm, and a device that responds appropriately to the recognized speech. This system varies in complexity along several dimensions. Miller states them as speed, speaking mode, vocabulary size, response time, background noise, quality and training (Miller and Walker, 1990).

One small problem with conversational interaction is you cannot even point to something you said a few lines ago in order to say it again. A second disadvantage of conversational interaction is that large scale structures are

difficult to manipulate as a whole. Noise tolerance is another problem where with systems susceptible to extraneous noise confusion, errors in recognition occur.

Systems need to have the capability of dropping down or moving up, to an appropriate interaction level on the basis of users' interactive behavior. One way to minimize user difficulty with voice based systems is to give control of the interactive process and the presentation of auditory information to the user. If he can slow, speed up, stop and repeat announcements, page backwards and forwards through the dialogue, the memory problem arising from the use of speech might be prevented. Another approach is to make the system sensitive to the user, by procedures like reaction time measurement, detection, etc. The user in difficulty can then be identified by the system itself and the dialog can be tailored accordingly. These two approaches are actually complimentary. The first involves providing control of facilities, and giving the user a helpful model of the system, so the he can take control of the dialogue. The second is system which develops a model of the user so that appropriate guidance can be given automatically. In this way the man-machine interaction moves towards a closer approximation of the way in which people converse together and control the dialogue process, on the basis of a model which shares knowledge about conversational usage (Monk, 1984).

Speech recognition systems do just what their name suggests; they recognize spoken words. Speech recognition detects words from speech. However, the recognition system does not analyze what those words mean. It only recognizes that they are words and what words they are. To be of any further use, these words must be passed on to higher level software for syntactic and semantic analysis. If the spoken words happen to be in the form of natural language, then they must be passed on to a natural language understanding system. It is important to understand that the "words" that make up a vocabulary to be recognized need not be words in the normal sense. Rather, speech recognition systems typically work by matching the acoustic pattern of an acoustic signal with the features of a stored template. According to Baecker and Buxton (1987) speech recognition systems vary along a number of dimensions:

Speaker dependent vs. independent: Speech recognition systems which are speaker independent are designed to recognize the speech of most speakers. Systems with limited vocabulary allows to enter data without logging on again each time a different person needs to use the system. However the speaker dependent systems are trained by the operator by repeating each word in the vocabulary several times and the system recognizes only the voice of him.

Dialects, accents or the language itself make no difference. Generally, systems with larger vocabularies are speaker dependent.

Continuous vs. dependent speech recognition: In continuous-speech recognition word endpoints are uncertain, since the user speaks at a normal conversational pace, and they can be determined by the aid of knowledge of the language syntax. In the discrete system word endpoints are determined by the periods of silence. According to Baecker and Buxton the factors that affect recognition system performance are:

- . user characteristics
- . enrollment
- . adaptive recognition algorithms
- . system feedback
- . error correction
- . environmental factors (Baecker and Buxton, 1987)

Voice recognition devices gather sound waves, remove unwanted noises, and compare the incoming signal against a template stored in memory. If the incoming sound is similar to what is on the template, then the word is recognized. If the sound is not similar enough to any stored template, then the system fails to recognize it.

The classes of errors that occur speech is presented to machines can be classified in four categories:

1. substitution errors: one word from the vocabulary is mistaken for another,
2. insertion errors: a word is reported that was not spoken,
3. deletion errors: a word that was spoken was not reported,
4. rejection errors: a word that is a legal item in the vocabulary is detected but not recognized (Baecker and Buxton, 1987).

Automatic output and input of speech from and to machines can be achieved using several different procedures, each with different advantages and disadvantages for a given application. First one is speech output using natural speech. Where recordings in sampled-data format can be stored in memory or on some rapid mass-storage device and played out as required. This strategy is appropriate for applications where a restricted set of high quantity words or phrases is required in providing commands or giving information. Second one is the speech output using syntactic speech. This is appropriate for applications with output of unpredictable messages from an unrestricted set, provided there

is no strict requirement that the speech sounds completely natural. The third procedure is Automatic Speech Recognition (ASR). Some form of speech recognition system is a prerequisite for a voice based user-machine interface. For now no system exists with capabilities near those of human listeners. However, by careful choice of constraints on the flexibility required for a given ASR application. Some useful working systems have been developed (Monk, 1984). Some of the issues that must be considered in their selection, evaluation and use are shown in table 4.2.

Type of speech:	isolated words, phrases, continuous speech
Number of talkers:	single talker, several designated talkers, unlimited
Type of talkers:	co-operative, casual, male, female, child
Environment:	sound-attenuating booth, computer room, public place
Channel to recogniser:	high quality microphone, high quality audio, noisy low-bandwidth telephone link
Type and amount of system training:	none, fixed training set, continuous
Vocabulary size:	small (<20 words), medium (<100 words), large
Speech format:	constrained text, free speech
Error tolerance:	high, low

Table 4.2. Considerations in the development, selection and evaluation of automatic speech recognition system. (Monk, 1984)

Speech is a discrete, single-channel, directional, well-known, semantically sophisticated system for the transmission of information. If properly implemented speech can reduce the need for the user to learn computer-programming, like languages and can provide an alternative to manual input systems.

4.10. Multi-Media Communication

Human-computer interaction techniques can include some peculiarities which enrich human dialogue and communicative possibilities with machine. These techniques have been listed by Baecker and Buxton as follows:

- . large display surfaces, such as provided by large format screens and video projection, that permit user to better manipulate spatial relationships and increase the amount of information.

- . voice input and output, and the use of non-speech audio output.
- . large capacity video storage with random access available by the compact disk technologies.
- . large scale digital data storage with random access available on the CD-ROOM technologies.
- . enhanced passing over and navigational tools, for example using spatial relationships.
- . the use of visual channel for input: a video camera which is used exploiting pattern recognition and machine vision techniques (Baecker and Buxton, 1987).

The use of these and other techniques can be combined in an multi-media environment. Variously termed multi-media, interactive media, or the new media, recent developments in electronic technology have made possible and accessible complicated technology. With this technique which include not only text, line art, and still images, but also sound, video sequences, computer graphics and computer-based animation, the mathematical content of a document can be symbolically and numerically manipulated. So, new results can be derived and different situations with different parameters can be simulated. Computer generated images can be explored by moving the eye point, changing the lighting conditions or using the model with algorithm. The user could also listen to recordings of the sounds or to view a visual, diagrammatic representation of those sounds and go on to compare this data with other species. The extend of information which can be made available is limited only by storage capacity and the resources of the creator.

Phillips states the most important potential of a document as “visualization”. Animations and still images in the original or a new document can be incorporated and transmitted electronically for viewing and analysis (Phillips, 1991).

Media View, for example, which is a system developed at Los Alamos National Laboratory on a Nexr workstation provides a generic infrastructure for creating and interacting with multimedia documents (Phillips, 1991). It is based on WYSIWYG. Figure 4.16. shows some of the potential components of a Media View document.

This document can have the following features:

- . any word or phrase in the volume can be retrieved and it can be accessed to any part of the document. Any piece of the document can be copied and pasted into a text editor or word processing program.

- . figures that support text can be viewed and if desired, selected for processing by another program.
- . an audio insert for listening is indicated by another icon style; for example it can be a part of the question and answer session. The sound can be arranged by a sound editor for incorporation in another multimedia document.
- . some notes can be made in the margin of the document by voice annotations or textual or graphical “stick-on” notes attached to the document.

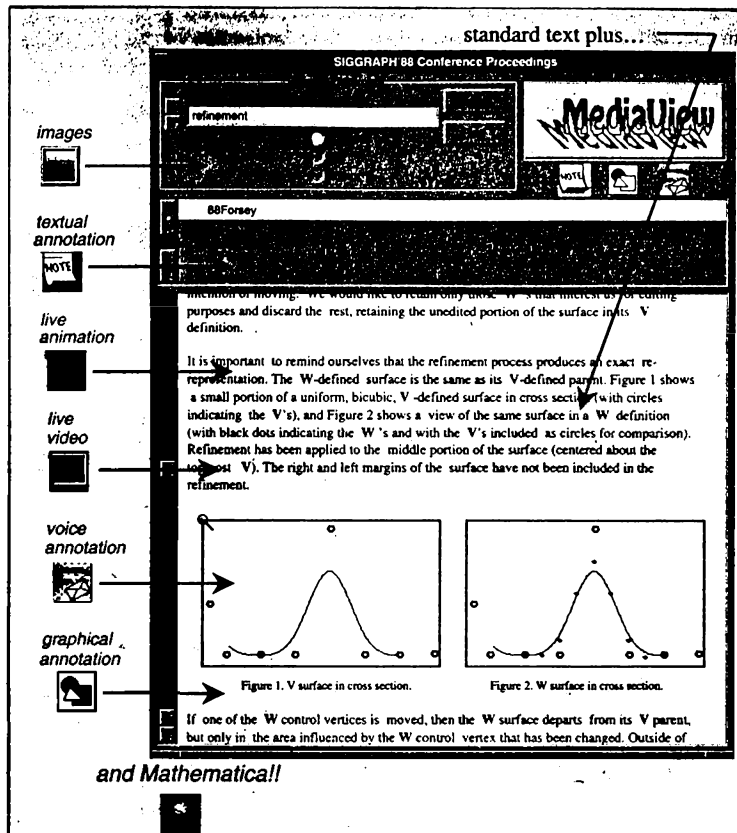


Figure 4.16. Components of a Media View document (Phillips, 1991)

Other available multimedia systems include Inter Media, developed at Brown University, and the Andrew Toolkit, from Carnegie Mellon University. Inter Media runs on an Apple Macintosh and uses the familiar look and feel of the Macintosh graphics user interface. The Andrew Toolkit runs on many Unix platforms, requiring only the X Window System (Phillips, 1991).

Another possibility of multi-media techniques for analysis and visualization is performing a simulation and using the data set animation facility to view the result. In addition to this, live video segments can be included in a document and facilitated by the video input and output capabilities. A click on the video

button will make the appearance of a video window and send the command to the appropriate device to begin video playback.

Recent developments in software marketing alter also the nature of what and how designers produce to a new degree, thus it challenges the role and the nature of graphic design, education and research. Gromala identifies the most significant feature of these authoring programs as “interactivity” which affects learning. Because it allows an intuitive interaction and offers information in textual, pictorial, animated, audio or video type. This learning environment give to the user the opportunity to perceive information in visual, aural, kinetic, or other sensory learning models. With these opportunities the user can contextualize information relating it to other types of content such as culture, society, polity, history or technique. In this respect users have options to create information and individualize them according to their own purposes (Gromala, 1992).

For the creation of these types of environments, the designer has to develop the conceptual framework of the complex array of information and technologies, to determine, construct and facilitate the possible ways where the user can work in this multi-media environment. The information must be comprehensible, accessible, and significant for the user; and he should not be forced to explore them.

On the other hand, installation of multi-media components into a document relies upon the ability to insert subclasses into the data structure of an instance class. The developer must implement methods for the cell's default behavior, such as how it should be highlighted, react to events, and draw itself. Most important, the developer must implement all subclasses of the class that give the multi-media components their distinctive behaviors (Phillips, 1991).

4.11. Virtual Reality

In some extreme cases a highly developed multi-media environment can be termed as “virtual reality”. In this three dimensional computer-generated environment where visual and audio capabilities exist, the user enters in a space with a high degree of simulation. This requires extensive computational capabilities. Combining real time graphics with 3D display systems is the first step toward achieving that may see as the ultimate goal of computer modeling - virtual reality- where the senses are immersed in a computer generated reality.

It could potentially fill the gap between “what you see” and “what you get” for three dimensional design. Gromala defines user experiences in this environment as “immediate feedback to his actions” (Gromala, 1992). Because of the user is isolated from all the outside conditions, he makes believe himself to be inside this computer-generated environment. With virtual reality, not only the can user see imaginary worlds (projected on screens right in front of his eyes, and with the illusion of infinite spaces) but now how can “feel” them (through specially constructed gloves) and (seem to) move around in them, as well (White, 1991).

In such a space there are no icon metaphors, keyboards, or specific language required from the user. According to the media, the user will soon all be involved in computer-generated worlds of incredible splendor. Wearing goggles that house miniature television screens, he will see these worlds in full stereoscopic vision, and wearing gloves wired with fiberoptics he will able to interact with these imaginary settings-flying through them simply by pointing a finger or grasping objects by making a fist with his “datagloves” (MacLeod, 1992).

If the eyes are surrounded by the virtual image of a computer-generated building, the mind is there as well. But how can the body follow: The mouse for virtual reality simulations is often referred to as 6D devices, since it measures movement in three dimensions and rotated about three possible axis. These devices include modified joysticks that are twisted and pushed, and magnetic trackers, such as an electronic glove, for instance, translate your pointing forward into moving forward towards a wall. Tapping the wall twice could represent a command to add a door. The system converts gestures and movements into electronic signals.

An additional element being added to datagloves is the sensation of touch or tactile feedback. If a glove could provide tactile feedback, the user feel an object in his hand as if he actually had something there. This more important for applications like remote robot hand control, but it also helps to maintain an equilibrium between what the eyes see and what the body feels (Yu, 1992).

A representation of a virtual building need not be limited to one person. Several people can strap on equipment and enter in the virtual world, seeing each other’s virtual bodies while the designer narrates a tour and music plays in the background.

Another possibility to input instructions is a microphone. Instead of keys on a keyboard there is already software that can execute a command or a series of commands from words spoken into a microphone. But, each user must prerecord

his or her own voice into the system. Considering the differences among voices and regional accents, this is an understandably difficult feature. Some have even lip reading as a simpler solution (Yu, 1992).

Drury believes that “virtual reality could revolutionize interactive electronic communication. It has the potential for removing a lot of the social boundaries that cause no end of trouble today”. He argues that “the next generations of virtual reality will allow people to create idealized constructs of themselves; and by adding a level of make-believe to reality, it becomes easier to communicate directly” (Drury, 1992).

A number of researchers and artists have been working in this relatively new field for several years. Nicole Stenger has been taking advantage of advances in virtual reality hardware to develop “Angels”, a “virtual reality movie”. Beverly Reiser occupies with Mandala Software’s complex interface to create interactive poems and stories. Working with both computer and video input, Michael Maimark develops multi-media and virtual reality projects like EAT -a “virtual dining environment” and moviemaps of Karlsruhe, Germany and Aspen, Colorado. Working on the borders of computer science and beyond, these virtual reality artists approach their work as both serious art and serious science (Haggerty, 1992). In addition to these fields, virtual reality has been used by the military as a method for training pilots example, by the doctors in training surgical techniques, or by the entertainment industries, etc. The most relevant and exciting field for architects that they have been using it to “walk through” or “fly by” simulations of spaces they design before the building is ever constructed (Gromala,1992). This “experiential prototyping” allows designers to see models simultaneously in true 3D, but without simultaneous viewing constraints (Gantz, 1992).

Virtual reality can be employed effectively in the applications defined above. But accurate and complete experience requirements must be assembled. Lanier states the toughest technical challenges of virtual reality as “connectivity” -that is, reading data from disparate systems and processing them in a virtual reality setting (Lanier, 1992). Cook identifies good virtual reality software as software that makes the best use of virtual reality’s unique capabilities and minimizes the weaknesses of the underlying hardware. According to him, it should let the user do something completely different in a way that either does not invite comparison with existing interfaces (Cook, 1992). Bryson asserted that faster computers, graphics, and access to data; higher resolution wide-field displays; more responsive input devices for 3D or higher, and interactive data exploration tools are needed. To evaluate visualization to research -not just use it for

display- it must be interactive (Bryson, 1992). In addition to this, Naimark defines the “camera” as the template for future virtual reality models. He suggests to virtual reality researchers to combine a knowledge of computers and computer graphics with an awareness of the conventions and capabilities of film (Naimark, 1992).

Virtual reality, the immersion of a user’s senses in computer-generated world, is often considered the far-fetched dream of eccentric computer buffs and science fiction fans. To present more practical functions for this technology, a two person virtual reality station for the creation of building prototypes running on PC computers is founded. The viewers donned helmets with separative view screens for each eye. As they turned or raised their heads, the screens smoothly reflected the shifting orientation. A joystick and a pointing wand were then manipulated to move walls, floors, and roofs in the “virtual world”, and to change the appearance of different surfaces. This rudimentary two person interaction is intended to represent an architect working with a client in the initial stages of a design project. This demonstration illustrates that the technological underpinnings for virtual reality exist, and the only necessary thing for practical use is time (Yu, 1992).

In fact, Autodesk, the worker of AutoCAD, has a virtual reality software project of its own called Cyberspace. The name “cyberspace” derives from science fiction author William Gibson’s books and stories which describe a decaying society where people plug themselves directly into global computer networks. All input and output is handled through direct neural connections into this hallucinatory “cyberspace”, bypassing the need for display devices and body suits (Yu, 1992).

The advantage or disadvantage of virtual reality is that it approaches reality. At what point it can be considered to be close enough to actual reality. Naimark asked what exactly is real time anyway, and began to analyze what makes a work virtual and real. “We have to be careful when we use the word real” he said. Realness, he explained, depends on both presence and interactivity. In computer graphics, we say something is *real* if it is of sufficient resolution. But *real* also belongs to the physical world. When do the two definitions coincide (Naimark, 1992)? It remains as an unclear but significant question for designers.

5. TOOLS FOR ARCHITECTURAL USER INTERFACE DESIGN

Although it is a common perspective that architectural design begins on a piece of paper and progresses through many stages until a blueprint, it does not mean that this is correct. However, some architect's view held in which they use a model that specify a way integrating the computer into the design process, is worth to pay attention. Negroponte's "architecture machine" must acknowledge and fit into the mentioned way of working that has initiated a search for hardware and software devices which would integrate; the machine with other, more traditional, aspects of computer-aided architectural design.

In practice, the world of computer-aided architectural design has grown away from this model. When "computer-aided architectural design" is translated as "computer-aided architectural draughting" it is recognized that the machine is used predominantly not as an aid to design, but as one of the specialized instruments of office practice: an instrument used to produce specific, dimensioned and annotated production drawings. Eastman's (1989) review of the state of the art goes so far as to suggest that idea of "integrated" CAD was misplaced.

All these suggest that there are two diametrically opposite approaches to CAD in architecture. The first stresses the central role that computing could play: it promotes a search for computing ways of doing the whole job and will be fully satisfied only when computer and design are inseparable. Brown and Horton (1992) call this the "strong" approach to CAD. The second, or "weak" approach is more pragmatic and seeks to automate only some parts of the design process which can be readily and naturally helped by computing. Both the "strong" and "weak" positions pose empirical questions about the architect and how the architect does his work. In computer terms these are questions about the user interface, about how architect and machine interact. In the strong CAD program they interact all the time from the first to the last point in the design sequence. In the weak model there are points where what is done by hand is to be presented to the machine and vice versa. The strong and weak approaches to CAD share a common set of problems to do with the way that designing and computing relate.

Drawing lines first and imaginary movements suggest that the tactile elements and graphic tools in drawing are the important parts of the experience. Repeating the pattern whilst drawing to screen would be possible in simple modeling type programs but would result in a number of irrelevant and confusing objects in a more sophisticated vector-based software. Furthermore, the utility of improved ergonomics -which are examined in the third section- should be asked. This might be an important and useful advance if talked about the gains obtained by making the work of a tracer quicker and more productive. For the architect such gains are likely to be less important since any time savings made will be spent by thinking time: intellectual work is not easily improved by time and motion study. However, there may be real gains to the architect in such an improved interface since it can speed up the way that ideas are transferred from mind to machine. For the experienced draughter sketching is a fast process which could be embedded by an unskillful machine interface. Work of the kind reported here by this thesis is valuable if it leads to a more *natural* use of the machine by architect, to the design of a better component.

In this respect, rising involvement of architects in computer-aided technology, the increasing need for communication and workability of information technology and the need for natural use of the machine are major factors that will require identification of some formal specifications, some standards and prototypes that will establish the intended goals.

5.1. Formal Specification of the Architectural User Interface Design

Since no stage in the transformation from problem definition and preliminary sketching phase to production of documents and working drawings, and construction the structure is thoroughly determined, design, in a creative sense, is involved at all architectural phases. In a computer-aided environment, it is moreover, dangerous to dissociate software design from design of the interface, because of unexpected interpretations of the specification and the danger that the system design may not coincide with the architect's intentions through constraints imposed by the interface. However, formal specification is a useful vehicle because it minimizes unnecessary dependencies, and allows transference of solutions from one problem domain to another. For this reason, formal specification of the user interface design should ideally remain open to modification throughout the architectural product development process.

A formal specification allows the designer to be rigorous about proving the internal consistency of that design, and about proving implementations with respect to design. Ideally, a formal specification should be explicit at the level of the primitive objects in its domain. The activity of design in architecture involves both the interpretation of a set of needs or desired performances in terms of a design, and the organization, composition, and transformation of combinations of (physical and symbolic) elements of buildings. Some architects set out a series of themes on and exercise in the syntax of architecture, concentrating on the nature and importance of the plan as an abstraction of architectural designs. Besides them some other architects carried out the essential formal elements of architecture as space and matter. All of them are equally important: the space between walls is to be designed as well as walls, the space between buildings as well as buildings. In a computer-aided environment the edges of architectural elements which will define these spaces can be manipulated: hard or permeable boundaries can be given to space. The operations on these elements may be transformations to and additions of plan, space and matter which are subject to ordering systems: axes, grids, proportions, the precedents set by formal specifications. For example when thinking about a project related especially for children or differently-abled people, the dimensions -such as maximum reaching height, minimum passage area, accessibility of the space, etc.- will be different than a project related to other people. In this respect, the interface with the machine may offer a guideline in forms of axes or grids that are derived from the preconditions. In another way, it may warn the designer with an error message -like "the child cannot reach to this point"-and evaluate the project according to these conditions. These contingencies will give the possibility to the architect to compose an infinite variety of forms of buildings.

In addition to this, color and texture should also be given to the matter. It can be made by different ways: In one way that is the designer who will define the model. The components of a texture or an element can be given by a firm and the designer can compose them in a manner as he wants. In the other way, the firm can give the different types and models of the element or texture. A draper for example can give the updated models and their costs in the form of modules which can be loaded to the system and then designer can choose among them. A more idealized system is that the firm can be dedicated to a network and load the updated models and their costs to this network and the designers in different environments can reach to them by this network. Alternatively, firms can also offer the life time, sound absorption, light reflecting and heat absorption values of the materials. They may be given by a model based system where these characteristics for example, may be simulated by the aid of color.

Another possibility of the interface may be the analysis of cost of the project according to the updated values; and when the designer changes the texture with another, the system should be able to change the estimated cost.

Formal specification is a valuable discipline and help to eliminate the ambiguity that is often present with informal specifications. The functionality of a system and the behavior of a system are essentially two subjects for a formal definition. Functionality can be defined as model-based or as algebraically. A model-based specification is more suited to systems for architectural design which uses drawings to represent physical objects and their configurations in space. But with simple set of operations algebraic specifications are also adequate. In a purely algebraic specification, the data type is implicit in the operations, while in a model-based specification it is expressed in the constructs of the model like sets and lists. In an architectural project, when thinking about the types of lamps for example; serial codes, some numeric values, etc. will not give enough information about the effect of this lamp when it is used in the project. Instead of the lists of numbers, the simulation of physical effects of these lamps will aid the architect more for decision making.

On the other hand, the behavior of a system may be defined as possible sequences of its states by model-based specifications. Expert systems for example, express operations as mappings between states, with pre-and-post-conditions. Baecker and Buxton state the importance of pre and post conditions with three reasons: 1. the checking of particular constraints is tied to particular operations, 2. the pre and post constraints for the operations act as a guide for the implementor, 3. they are used to prove that the specification and implementation are correct. (Baecker and Buxton, 1987) These arguments can be illustrated for the design of two-dimensional layouts of rectangles for example, that may be adapted to different domains. The expert system should be able to systematically enumerate alternative solutions with interesting tradeoffs, taking into accounts broad spectrum of criteria and practical concerns. The domain knowledge may be incorporated as test rules, and may be adapted to some domains, for example to the remodeling of residential kitchens:

1. The preprocessor may accept from the architect a problem statement consisting of a *context description* and a *list of objects* to be allocated. For the present domain, the context typically will consist of walls, windows and doors forming the boundary of the kitchen; to these may be added existing fixtures such as a radiators. All of these objects may be considered rectangles, and their shape and position may be specified by architect through their corner coordinates. From these specifications, the preprocessor may generate a

configuration that represents the given arrangement of context elements; it may function as the starting configuration for the search.

2. **Tester:** The rectangles that have to be allocated can be called *design objects* to distinguish them from the rectangles that make up a context. The design objects may be allocated as: work area, sink, refrigerator, range, work counter. After that, some conditions for each object may be described to lead to the criticism. These conditions may be such as “back of refrigerator cannot be placed against a wall”, “sink should have space on either side for a work counter” or “work area should be accessible from the door or opening that leads to dining area” etc. The complete set of test rules leads to the alternatives.

3. **Post-processor:** The post-processor invoked by the control strategy, should refine solutions.

The inference mechanism controls the strategy of the system. It should be capable of handling both goal driven and data driven strategies. The sequence of strategy should be such that only the information required is asked and the architect should be able to provide information at as high a level as he is capable. It must have an efficient selection mechanism and the capability of undoing any portion processed if required. It should distinguish between the inability to arrive at a particular consequence and that consequence failing. The explanation facility must be able to explain why the system needs a particular piece of information; how it arrived at a particular conclusion; which conclusions failed and why, and why a particular conclusion was not reached.

The user interface provides access to the architectural operations, that is it generates a model within the user interprets the effects of the operation. The architect’s interpretation of the operation is therefore strongly modified by its interface. So interface constraints on their design, since unless such effects are taken into account, what the architect perceives may be at odds with the system designer’s intentions.

5.2. Need for Prototypes of the Architectural User Interface Design

One difficulty in designing interactive systems is that the designer and the user may not have a clear idea of what the system will look like when it is done. So several works must exist. First, there must be good evaluation techniques so that the strengths and weaknesses of the design can be determined. Second,

there must be rapid prototyping tools that make it easy to try out new ideas. As prototype versions become available, testing can be more elaborate. These preliminary tests help build confidence that the acceptance test can be satisfied when the implementation is complete. Rapid prototyping has the advantages that ideas can be tested immediately while they are still fresh in mind, and that users and designers get immediate feedback, thus -rewarding their sense of participation in the design.

More recently, emphasis has been placed on the importance of architectural design knowledge as the origin of the intuitive and irrational decisions made by the interface designer. The availability of knowledge to extend the design space in a computer based environment, coupled with effective prototypes for searching the design space for alternative solutions, has become the basis of a promising new direction in computer-aided architectural design.

For many interactive systems, the optimal design and prototyping approach is to perform the complete interaction design on a major subsystem. The interaction design for that subsystem then becomes the model for other subsystems to be used by other users having similar skills, to achieve a consistent user interface throughout the system. In a computer-aided architectural environment, architects' experts can be simulated by a set of prototypes in which each system has a special domain. In this respect the main system will not be too loaded, and each subsystem will be expert in its context and customize for the needs of a particular area. A coordination facility can integrate these subsystems such that the computer system as a whole will participate as an assistant during the development of a design solution. In functional terms, such an assistance facility could be referred to as a design advisor. The design advisor in the prototype interprets a drawing as it is being made by a designer working in a CAD environment. It reacts in real-time to monitor the evolving floor plan from the viewpoints of experts in the domains of access, climate, cost, lighting, sound, structure, etc.

One of these systems may be a prototype for sunlight design for example. The interface may be shown to the architect with the sunlit room under investigation which may be presented as five rectangular black areas, representing the floor and the four walls folded flat into the plane of the computer screen. Building orientation, latitude and times of day and year should be selected by architect using familiar computer techniques such as scroll bar and click-dragging. All these variables should be easily accessed and be visible on the interface to facilitate modification at any time. The final element of the interface may be the simple graphical toolbox which allows the architect to draw on the walls or floor

of the room. The tool should be able to be used in two principal ways. The architect should be able to draw any desired window shape on the room wall and the tool immediately shows the areas of sunlight cast for the times, location and orientation specified. Alternatively, and more interestingly in the context of the present prototype, the architect should be able to sketch a desired patch of sunlight to which the tool responds by showing the shape and location of the window(s) needed to cast a patch of sunlight of this particular shape. This clearly gives the architect access to a real design tool; window locations should be able to be chosen to give a desired performance characteristic.

This proposed application may be only a prototype. It may be applicable to complex room shapes and fine control over orientation. The principal is establishing, and that is the important issue. However, the tool is actually aid to the architect's understanding of sunlight penetration into the buildings.

An obvious development would be to link the kind of application described here to a drawing-modelling CAD package, the aim being to give the architect interactive feedback about the technical performance of the building as it is being developed in the CAD application. For example the system may give some ideas to the architect about the electromagnetic smoke of the room, its ionization, mineral and non-mineral balance, etc. -that means the effects that cannot be seen easily by pure eye. It can analyze the conditions of the room -such as light, heat, effect of sun, etc.- and suggest an appropriate plant for example, for the current room. A proposal for the system in making these suggestions may be for example the simulation of color spectrum.

Another prototype may be introducing the time dimension. A serious drawback may be to optimize the system's speed as an administrative aid. A number of the key views may be colored with a color paint program. This may be a slide show utility, as well as the ability to cycle colors through time. It may be easy to create the appearance of night gradually falling over the building and lights coming on, as well as the reflections of passing clouds in the exterior mirror glass. The computer screen may introduce the passage of time and the effects of moving light and shadow.

In addition, to express ideas in architectural language, current systems must model objects. More powerful systems can add other features such as texture mapping and can be faster and smoother. Yu (1992) defines the ways that a computer can prepare the prototype of an architectural design element with the following perceived dimensions:

- . Computers can create holographic projections, but this “synthetic dynamic holography” requires tremendous computer power. Systems under development at such research sites utilize supercomputers to provide the number crunching required, and even then only small and short animations are made.
- . Cylindrical displays that display an image from any angle in 3D: these “true volume images” rotate a 2D matrix of light emitting diodes or similar light sources inside a clear cylinder to create a 3D image.
- . CAD systems can produce models with processes often used in computer-aided manufacturing for product design. This “solid prototyping” generates model of 3D images by using lasers to fuse metal powders or light sensitive plastics, forming a physical model. However this is difficult to implement when an object has many surfaces inside other surfaces as in the simplest house.
- . Polarization system which allows to view the object from different angles and gives a sense of three dimensional.
- . Miniature television screens, goggles and gloves which make believe the user to be inside of a computer-generated environment: virtual reality.

In conclusion, the need for prototypes of user interfaces can be summarized as follows:

1. It enables the user to evaluate the interface in practice and to suggest changes to the interface.
2. It enables the developer to evaluate user performance with the interface and to modify it so as to minimize user errors and improve user satisfaction.
3. It facilitates experimentation with a number of alternative interfaces and modification of interfaces.
4. It gives the user a more immediate sense of the proposed system and thereby encourages users to think more carefully about the needed and definable characteristics of the system.
5. It reduces the likelihood of project failure (Preece and Keller, 1990).

5. 3. Standardization of the Architectural User Interface Design

The increasing need for intercommunication and information technology affects the developments in standardization. The single user becomes an important factor in the standardization process. Studies have proved that the character set, graphics, languages, software application, screen and board are the issues with the highest percentages of standards produced.

User Interface Management System is a software system that supports the presentation of data on the screen and accepts users reactions, it also manages all user interactions with the computer independent of the application it is running. User Interface Development System is an integrated set of tools used by programmers for the user interface design and development. A major advantage of this system is that it helps separate the design of the user interface from its implementation.

In computer-aided architectural design, standards in information of structural conceptions onto the constructs of a given system are very important. Effective use of a draughting system to construct a plan or elevation, for example often depends on recognition of repeating parts and the structure of the repetition, followed by shrewd use of copy-and-transform operations. During this work, the architect should have to know where he is, how did he get here, where can he go and how to get there. So, the site and mode properties where the architect works should be defined in a standard manner.

On the other hand, effective use of a surface modeler may depend on the ability to see building as a collection of translationally and rotationally swept profiles. Effective use of a solid modeler may depend on the capacity to see complex shapes as unions, intersections or differences of simpler ones. And, effective use of an interactive walk-through system may depend on the ability of selection and identification mechanism. For this reason a proposal may be as follows: Clicking on the left or right side of the screen may turn one in that direction. Clicking in the centre may move one forward. Clicking on a door, skylight or window, may jump one through it. Such standards can be developed through practice in construction of representations of conceptually demanding objects, such as major works in architecture.

Early CAD systems provided such primitive interfaces (e.g. keyboard commands) that they virtually eliminated the need for hand-eye skills. But as architectural systems provide broadband interfaces, and are able to interpret complex gestures and to provide much richer forms of feedback, the need for hand-eye skills becomes important. Skilled architects of CAAD systems should be able to execute complex gesture patterns, attuned to all the nuances of the feedback that flows from them, and ready to make subtle adjustments in response. So for architects who express their ideas graphically, the most important features of the graphical user interface is the device independence. In addition to this, the *look and feel* composition, and the data interchange possibilities are very important. In this respect, all software packages of a dominating system which should offer these possibilities in all different micro's

operating system environments should have the same feature, and operate in essentially the same manner. So once an architect learns one application, they should have learned the basics of them all. Because of this consistency, architects can copy the work they have done in one application and paste it into another. This situation will let the architect devote more time to getting work done, and less to memorizing difficult computer commands and provides the possibility to influence the finalization of the standards in a way favorable for the interests of the architects.

Standardization bodies should be encouraged to bring together the different standardization activities concerning the user interface areas that are spread across different sectors and form standardization sectors dedicated to the user interface area. But however integrated stand-alone CAAD systems that store all the required information in a single file sometimes require the architect to commit himself to decisions that he is not yet ready to take. Typically a system might require the designer to specify all components in a library before a single line can be drawn. Whereas architects prefer to leave themselves room to manoeuvre by not being too definitive with their ideas at the beginning. By slowly building up design information from sketch doodles to sketch plans, through models and general arrangement plan to detailed schedules and specifications, there should be a better chance that the right decisions are taken at the appropriate time.

Regional and local deviations, such as different date, time and addresses formats, currency symbols, punctuation and decimal point marks, and also more general ones such as character sets and collating sequences, appear to play an important role on the user perception of the computer system. The standardization bodies should be encouraged therefore to reflect more on the interests of local user groups with different cultural traditions, languages, alphabets like standards on character sets, keyboards, command languages and presentation formats.

6. CONCLUSION: EVALUATION OF THE INTERFACE FOR CAAD

Very often the software instruments compel the architect to organize his own work according to the logic and the methodology offered by the software used, which sometimes do not coincide with his working method. The instruments which aid design, should make it possible for the architect, even he is not an expert, to describe, access and control the desired aspects of the buildings in the process of definition in language which is simple and adequate to his own working method. So, it can be argued that, the nature and power of the conceptual tools available to the designer determine in no small measure what he can conceive and accomplish. And conversely, the limitations of methods will be expressed as limitations of the design. For this reason it should be accepted that, the aspects of architectural design activity should be at the basis of the creation of software systems aimed at collaborating with the architect in this type of activity. The architectural product will need to be evaluated during the development process. This evaluation should be intended to establish that the usability goals have been achieved, and redesign of the user interface may be necessary. First of all evaluation of architect-computer interface must address all interfaces between the architect and the computer, and not only just the display interface. Secondly, an evaluation, to be effective in producing an improved system, must begin earlier than when the program is completed.

When thinking about the role of architects, we express that they represent physical objects and their configuration in space. Since they prefer to describe their designs through drawings as well as specifications, the system should allow a familiar language -the picture- where the designer can talk to the machine graphically and the machine can graphically respond in turn. This graphical representation will allow for the dynamic manipulation of geometry, which will provide the direct interaction between the architect and the object form. This is a valuable feature of the craft process that may be potentially recovered by CAAD.

There are, however some problems attached to the integration of computer-aided architectural design with application programs. The first problem is that,

usually, the CAAD programs contain graphic databases in a form not readily accessible from the application programs. Another problem is that the typical CAAD system today is a general purpose graphical system. It understands lines and circles, text, and perhaps raster images. In some cases, it understands three-dimensional forms -planes, surfaces and solids. It does not know anything about buildings or architecture. All the meaning mapped from this graphical representation is derived by architects using their knowledge and experience. These systems have low intelligence but are highly flexible. The same software is equally adept at drawing a building, a landscape or a ship.

A proposal especially for a computer-aided architectural design may be an approach on a different level. The elements that the system deals with may be slabs and walls, doors and windows and roofs and roof-lights. This will be more intelligent to an architect; but it can be less flexible at the same time. There may be general purpose objects intended for furniture and equipment, but how about columns, beams, rafters, pipes, ducts, piles and all the hundreds of other elements that might occur in a building? In general it will be harder to deal with them than in a general purpose system.

Another approach which treats buildings as sculptural volumes, knows about their subdivisions into floors and zones, and knows the location of cores and circulation. Additionally it may deal with occupants -people, plant and processes. This will be a more functional model than the others, and at the same time hard to combine with them. Yet it will be better while a building concept is being formed, and for cost planning and environmental analysis.

Another proposal may be the introducing the fourth dimension -the time- into the computer-aided design process. With the virtual reality technique architects will soon be able to walk their clients through their proposed design in three dimensions. However, they have some technical problems -such as the low resolution of the head-mounted displays, the lag time between moving the eyes and changing of the display, and the limited range of the tracking devices. Virtual reality gives a whole new real estate to be developed. In other words, in the future architects may find themselves designing virtual environments rather than buildings, as these environments may be radically different from anything they have ever designed before. When thinking of virtual reality as a place to go, the kinds of things that cannot be done in architect-designed buildings today can be explored in it. The process of designing in this environment may in itself be a radically different approach to architecture. But if virtual reality has a real effect on architecture it must offer improved hardware; because in its most common form it is really just graphics. Until the information that defines a

building is organized in a coherent, complete, and structured form, virtual reality is just another means for making interesting pictures, and as such, is a dead-end for the real business of using computers to describe the design of buildings. Rather than another rendering device, architects need tools that unite all the various aspects of design -plans, sections, specifications, estimates, and code checks- into a single database.

Gradually virtual reality is not also a panacea for either the world's or architecture's problems. There are technical, conceptual, and even ethical questions that must be answered before it can be used in any meaningful way. And these questions will not answer themselves. They demand that architects re-examine both the way they design buildings and their social responsibility to ensure that new technologies are used widely. They show promise, but its growing pains are sufficiently disturbing to require that architects have to pay careful attention to their continued development.

If what is required in architecture is a means of modeling so that a design may be presented graphically, the elements modeled should be recognized and understood by the mechanism before carrying out an operation. The machine must further be able to communicate, access knowledge, discern changes in meaning brought by changes in context, and reason i.e. make inferences: arrive at conclusions and advise, explain, and/or justify the reasoning: revise the reasoning and learn. In order for a computer system to understand a graphical representation and make inferences it is necessary to incorporate sufficient knowledge about the underlying physical objects for the system to carry out the mapping between the syntactical representations and the required semantics. Such a degree of understanding and discerning changes can be introduced in a system through explicit knowledge representation, intelligence, inference mechanism and explanation facility. To do this, the system must have a sophisticated set of sensors, effectors and processors to view the real world directly and indirectly. What makes this behavior unique and particularly difficult to emulate machines is its extreme dependence on context: time, locality, culture, mood, and so forth.

As a result, the future role of the CAAD system designer should be know how to combine them so as to achieve both flexibility, intelligence and time without excessive complexity.

The most likely route may be to construct a system in layers. This will treat the system as composed of a hierarchy of objects, with each level being more specialized and intelligent. At the root there will be general purpose object, from

which ascend user interface objects such as windows, icons and menus, process objects such as commands and undo stacks and the familiar graphical objects of the general purpose CAAD system.

Intelligent building objects (like slabs and walls), will be at a higher level, and will know how to represent themselves in the more primitive terms of graphical objects in two-dimensions and three-dimensions, or even as text. They will also contain their own rules for intelligent behavior, for example a window will know that it must make an appropriate opening in a wall and a wall will know that it must extend upwards to the ceiling or to the structural slab. To retain generality it will be essential that these rules be easy to formulate and change according to the needs of a project. This can be realized by the use of logic programming and intelligent knowledge-based systems.

The storage of a model constructed like this, in such a way that it is accessible to many different applications, presents an important problem. The techniques of clipboard, scrapbook and import and export procedures may not be adequate, and anyway cannot be used for substantial models. Current approach may be to suggest a database to isolate the data from its applications and allow for parallel access from all members of the team. However, relational databases are not a good match with the object-oriented philosophy, and something new will be needed. Some hope may be that current research should be related with the entity modeling, engineering databases and object-oriented databases which should be converged into a post-relational solution.

If the objects in system have variable intelligent behavior, then the way that a designer interacts with them should be similarly variable. Thus the user interface must be customizable. But, there is still no satisfactory way of dealing with the "feel" -the resulting behavior of the interface and the objects it controls- other than a textual programming language.

In summary, tomorrow's computer-aided architectural design systems will combine graphical representation with direct interaction between the architect and the object form; knowledge representation, intelligence, inference mechanism and explanation facility which will evaluate the building elements modeled, arrive at conclusions, and explain, justify and revise the reasoning; and the fourth dimension -the time- which will provide walk through the proposed design in three dimensions; and achieve both flexibility and customizable user interface.

REFERENCES

- Arnheim, R. 1969. Visual Thinking. Berkeley: University of California Press.
- Baecker, M., and Buxton, W. A. S. ed. 1987. Readings in Human-Computer Interaction: A Multidisciplinary Approach. California: Morgan Kaufmann Publishers, Inc.
- Baecker, R. M., and Marcus, A. 1989. Human Factors and Typography for More Readable Programs. USA: Addison-Wesley Publishing Company, ACM Press.
- Barnard, P. J. et al. 1981. "Consistency and Compatibility in Human-Computer Dialogue." International Journal of Man-Machine Studies. 15: 87-134.
- Botterill, J. H. 1982. "The Design Rationale of the System/38 User Interface." IBM Systems Journal 21: 384-423.
- Boulay, B. D. et al. 1981. "The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices." International Journal of Man-Machine Studies 14: 237-249.
- Bowman, W. J. 1968. Graphic Communication. New York: John Wiley and Sons.
- Brown, A. G. P., and Horton, F. 1992. "Computer Aids for Design Development." Computers In Architecture. Ed. P. François. UK: Longman Group Limited. 15-24.
- Brown, J. S. 1986. "From Cognitive to Social Ergonomics and Beyond." User Centered System Design. Ed. D. A. Norman and S. W. Draper. London: Lawrence Erlbaum Associates. 474-475.
- Bryson, S. 1992. "VR and Fluid Flows" IEEE Computer Graphics and Application 12: 18.
- Card, S. K., Pavel, M., and Farrell, J. E. 1984. "Window-Based Computer Dialogues." Interact '84. 355-359.
- Carroll, J. M. 1982. "Learning, Using and Designing Command Paradigms." Human Learning 1: 31-62.
- , 1987. "The Adventure of Getting to Know a Computer." Readings in Human-Computer Interaction: A Multidisciplinary Approach. Ed. R. M. Baecker and W. S. Buxton. California: Morgan Kaufmann Publishers, Inc. 639-648.

- Chi, U. H. 1985. "Formal Specifications of User Interfaces: A Comparison and Evaluation of Four Axiomatic Approaches." IEEE Transactions on Software Engineering SE-11: 671-685.
- Cockton, G. 1990. "Designing Abstractions for Communication Control." Formal Methods in Human-Computer Communication. Ed. M. Harrison and H. Thimbley. Cambridge: Cambridge University Press. 240-242.
- Cook, R. 1992. "Serious." Computer Graphics World May: 40-48.
- Coons, S. A. 1963. "An Outline of the Requirements for a Computer-Aided Design System." AFIPS Conference Proceedings 23: 299-304.
- Cross, N., and MAVER T. W. 1973. "Computer Aids for Design Participation." Architectural design May" 46-50.
- Cross, N. 1977. The Automated Architect. London: Pion Limited.
- Davis, R. M. 1966. "Man-Machine Communication." Annual Review of Information Science and Technology. Ed. C. A. Cuadra. New York : Interscience. 1: 221-254.
- Dean, M. 1982. "How A Computer Should Talk to A People." IBM Journal 21: 424-453.
- Dondis, D.A. 1973. A Primer of Visual Literacy. Cambridge, Mass: MIT Press.
- Draper, S. W. 1986. "Display Managers." User Centered System Design. Ed. D. Norman and S. Draper. London: Lawrence Erlbaum Assoc. 339-352.
- Drury, M. 1992. "Virtually Endless Possibilities." Compuserve Magazine. 2:6.
- Durgun, F. B., and Özgüç B. 1990. "Architectural Sketch Recognition." Architectural Science Review 33: 3-16.
- Eastman, C. 1989. Architectural CAD: A Ten Year Assessment of the State of the Art Computer-Aided Design. London: Butterwoths.
- Fallon, K. 1990. "Beyond Computers as Pencils." Architectural Record November: 28-31.
- Fano, R. M., and Corbato, F. J. 1966. "Time Sharing on Computers." Scientific American 214: 129-140.
- Foley, D. J. Wallace, V. L. and Chan, P. 1990. "The Human Factors of Computer Graphics Interaction Techniques." Human-Computer Interaction. Ed. J. Preece and L. Keller. UK: Prentice Hall International Ltd. 67-121.
- Furnas, G. W. et al. 1987. "The Vocabulary Problem in Human-System Communication." Communication of the ACM 30: 948-955.
- Gaines, B. R., and Mildred, L. G. S. 1986a. "Foundations of Dialogue Engineering: the Development of Human-Computer Interaction." International Journal of Man-Machine Studies Part 1, 24: 1-27.

- _ _ _ , 1986b. "Foundations of Dialogue Engineering: The Development of Human-Computer Interaction." International Journal of Man-Machine Studies Part II, 24: 101-123.
- Gantz, J. A 1992. "Virtual Market." Computer Graphics World May : 27-28.
- Gittins, D. T. et al. 1984. "An Icon-Driven End-User Interface to Unix." International Journal of Man-Machine Studies 21: 451-461.
- Green M. A. 1979. "Graphical Input programming System." Diss. Toronto U.
- Gromala, D. J. 1992. "Multi-Media in Graphic Design." Academic Computing In Macintosh Environment III. Eskişehir: Anadolu University Press. 1-9.
- Haggerty, M. 1992. "Serious Lunacy: Art in Visual Worlds." IEEE Computer Graphics and Application 12: 5-7.
- Hammond, N., and Barnard P. 1984. "Dialogue Design: Characteristics of User Knowledge." Fundamentals of Human-Computer Interaction. Ed. A. Monk. London: Academic Press, Inc. 130-143.
- Hansen, W. J. 1971. "User Engineering Principles for Interactive Systems." AFIPS Conferences on Proceedings 39 :523-532.
- Henry, T. R., and Hudson, S. E. 1990. "Multidimensional Icons." ACM Transactions on Graphics 9: 133-137.
- Hersh, H. N. 1982. "Icon Assessment." London: Digital Equipment Corp., Corporate Research, unpublished document.
- Hix, D., and Schulman, R. S. 1991. "Human-Computer Interface Development Tools." Communications of the ACM 34: 74-87.
- Hooper, K. 1986. "Architectural Design: An Analogy." User Centered System Design. Ed. D. A. Norman and S. W. Draper. London: Lawrence Erlbaum Associates. 10-13.
- Hornbuckle, G. D. 1967. "The Computer Graphics/User Interface." IEEE Transactions on Human Factors in Electronics HFE-8:17-22.
- Huggins, W. H., and Eentwisle, D. R. 1974. Iconic Communication: An Annotated Bibliography. Baltimore: The Johns Hopkins Press, Md.
- Johnson, T. E. 1963. "Sketchpad III.: Three-Dimensional Graphical Communication with a Digital Computer." AFIPS Conference Proceedings 23:347-353.
- Kammersgaard, J. 1990. "Four Different Perspectives on Human-Computer Interaction." Human-Computer Interaction. Ed. J. Preece and L. Keller. UK: Prentice Hall International Ltd. 44.
- Kenzie, J. Mc. 1988. "Guidelines and Principles of Interface Design." Designing End User Interfaces. England: State of the Art Report, Pergamon Infotech Limited. 82.

- Kiger, J. I. 1984. "The Depth/Breath Trade-off in the Design of Menu-Driven User Interfaces." International Journal of Man-Machine Studies 20: 201-213.
- Lanier, J. 1992. "A Virtual Market." Computer Graphics World May: 27-28.
- Licklider, J. C. R. 1960. "Man-Computer Symbiosis." IRE Transactions on Human Factors in Electronics HFE 1: 4-11.
- _____, 1968. "Man-Computer Communication." Annual Review of Information Science and Technology. Ed. C. A. Cuadra. New York: Interscience. 3: 201-240.
- Licklider, J.C.R., and CLARK, W. E. 1962. "On-Line Man-Computer Communication." AFIPS Conference Proceedings 21: 113-128.
- Lodding, K. N. 1983. "Iconic Interfacing." IEEE Computer Graphics and Applications March/April: 11-20.
- Macleod, D. 1992. "Computers: Virtual Reality." Progressive Architecture 73: 55-56.
- Maguire, M. 1982. "An Evaluation of Published Recommendations on the Design of Man-Computer Dialogues." International Journal of Man-Machine Studies 16: 237-261.
- Mantei, M., and Haskell, N. 1983. "Autobiography of a First-time Discretionary Microcomputer User." CHI' 83 Proceedings 2: 286-290.
- Marcus, A. 1983. "Graphic Design for Computer Graphics." IEEE Computer Graphics and Application 3: 63-70.
- Miller, R. K., and Walker, T. C. 1990. Natural Language and Voice Processing. Lilburn The Fairmont Press, Inc.
- Monk, A. ed. 1984. Fundamentals of Human-Computer Interaction. London: Academic Press, Inc.
- Moran, T. P. 1981. "The Command Language Grammar: A presentation for the User Interface of Interactive Computer Systems." International Journal of Man-Machine Studies 15: 3-50.
- Naimark, M. 1992. "Serious Lunacy: Art in Visual Worlds." IEEE Computer Graphics and Application 12: 5-7.
- Negroponte, N. 1970. The Architecture Machine. Cambridge, Massachusetts: MIT Press.
- Nelson, T. H. 1987. "Human Factors of Interactive Software." Designing the User Interface: Strategies for Effective Human-Computer Interaction. Ed. B. Shneiderman. USA: Addison-Wesley Publishing Company. 9.
- Nickerson, R. 1969. "Man-Computer Interaction, A Challenge for Human Factors Research." Ergonomics 12: 501-517.

- Nickerson, R. 1969. "Man-Computer Interaction, A Challenge for Human Factors Research." Ergonomics 12: 501-517.
- _____, 1981. "Why Interactive Computer Systems are Sometimes not Used by People Who Might Benefit From Them." International Journal of Man-Machine Studies 15: 469-483.
- Nievergelt, J., and Weydert J. 1987. "Sites, Modes and Trials: Telling the User of an Interactive System Where He Is, What He Can Do, and How to Get to Places." Readings in Human-Computer Interaction: A Multidisciplinary Approach. Ed. R. M. Baecker and W. A. S. Buxton. California: Morgan Kaufmann Publishers, Inc. 438-441.
- Norman, A. D. 1986. "Cognitive Engineering." User Centered System Design. Ed. A. D. Norman and S.W. Draper. London: Lawrence Erlbaum Associates. 41-54.
- Phillips, R. 1991. "An Interpersonal Multimedia Visualization System." IEEE Computer Graphics and Applications 11: 20-27.
- Preece, J. and Keller, L. ed. 1990. Human-Computer Interaction. UK: Prentice Hall International Ltd.
- Reichman, R. 1986. "Communication Paradigms for a Window System." User Centered System Design. Ed. D. A. Norman and S. W. Draper. London: Lawrence Erlbaum Associates. 285-313.
- Riley, M. S. 1986. "User Understanding." User Centered System Design. Ed. D. A. Norman and S. W. Draper. London: Lawrence Erlbaum Associates. 161-169.
- Rissland, E. L. 1984. "Ingredients of Intelligent User Interfaces." International Journal of Man-Machine Studies 21: 377-388.
- Rosenman, M. A. et. al. 1988. "Solarexpert: an Expert System for Passive Solar Energy Design in Housing." People and Technology: Sun, Climate and Building. Ed. S. V. Szokolay. Brisbane: Proceeding of the Joint Conference of Anzasca and Anzsas, University of Queensland. 121-128.
- Ross, D. T., and Rodriguez, J. E. 1963. "Theoretical Foundations for the Computer-Aided Design System." AFIPS Conference Proceedings 23: 305-322.
- Ryan, D. L. 1986. Modern Graphic Communications, A CAD Approach. USA: Prentice Hall.
- Salvendy, G. ed. 1986. Human-Computer Interaction. New York: Elsevier Science Publishing Company Inc.
- Schneider, W. 1985. "Training High-Performance Skills: Fallacies and Guidelines." Human Factors 27: 285-300.
- Shackel, B. 1969. "Man-Computer Interaction: the Contribution of the Human Sciences." IEEE Transactions on Man-Machine Systems 10: 149-163.

- Shneiderman, B. 1980. Software Psychology: Human Factors in Computer and Information Sciences. Cambridge, MA: Wintrop Publishers.
- , 1982. "The Future of Interactive Systems and the Emergence of Direct Manipulation." Behavior and Information Technology 1: 237-256.
- , ed. 1987. Designing the User Interface: Strategies for Effective Human-Computer Interaction. USA: Addison-Wesley Publishing Company. 135-177.
- Stotz, R. 1963. "Man-Machine Console Facilities for Computer-Aided Design." AFIPS Conference Proceedings 23: 323-328.
- Sutherland, I. 1963. "Sketchpad." Siggraph Video Review. New York: ACM.
- Sutherland, W. R., Forgie, J. W., and Morello, M. V. 1969. "Graphics in Time-Sharing." A Summary of the TX-2 Experience 34: 629-636.
- Szekeley, P. 1987. "Separating the User Interface from the Functionality of Application Programs." ACM SIGCHI Bulletin 18: 45-46.
- White, R. 1991. "Into the New World." Artweek 22: 15.
- Whitehead, A. 1984. Human Factors Aspects of Pointing as an Input Technique in Interactive Computer Systems. London: Ergonomic Unit, University College.
- Whitfield, D. 1967. "Human Skills as a Determinate of Allocation of Function." The Human Operator in Complex Systems. Eds. W. T. Singleton, R. S. Easterby, and D. Whitfield. London: Taylor and Francis. 54-60.
- Winfield, I. 1986. Human Resources and Computing. London: William Heinemann Ltd. 52-60.
- Wright P. 1988. "Communicating with the User." Designing End-User Interfaces. England: State of the Art Report, Pergamon Infotech Limited. 123-129.
- Yu, L. 1992. "Virtual Reality Demonstrated." Progressive Architecture 73: 30.